Developer Guide | Langchain

Skip to main content LangChainDocsUse casesIntegrationsAPIMoreCommunityTutorialsContributingAlso by LangChainChat our docsLangSmithLangChain HubLangServePython DocsSearchCTRLKDeveloper GuideContributing to LangChain Hi there! Thank you for being interested in contributing to LangChain.

As an open source project in a rapidly developing field, we are extremely open

to contributions, whether it be in the form of a new feature, improved infra, or better documentation.To contribute to this project, please follow a "fork and pull request" workflow. Please do not try to push directly to this repo unless you are a maintainer.Quick LinksNot sure what to work on?If you are not sure what to work on, we have a few suggestions:Look at the issues with the help wanted label. These are issues that we think are good targets for contributors. If you are interested in working on one of these, please comment on the issue so that we can assign it to you. And if you have any questions let us know, we're happy to guide you!At the moment our main focus is reaching parity with the Python version for features and base functionality. If you are interested in working on a specific integration or feature, please let

us know and we can help you get started.New abstractionsWe aim to keep the same APIs between the Python and JS versions of LangChain, where possible. As such we ask that if you have an idea for a new abstraction, please open an issue first to discuss it. This will help us make sure that the API is consistent across both versions. If you're not sure what to work on, we recommend looking at the links above first.Want to add a specific integration?LangChain supports several different types of integrations with third-party providers and frameworks, including LLM providers (e.g. OpenAI), vector stores (e.g. FAISS), document loaders (e.g. Apify) persistent message history stores (e.g. Redis), and more.We welcome such contributions, but ask that you read our dedicated integration contribution guide for specific details and patterns to consider before opening a pull request.Want to add a feature that's already in Python?If you're interested in contributing a feature that's already in the LangChain Python repo and you'd like some help getting started, you can try pasting code snippets and classes into the LangChain Python to JS translator.It's a chat interface wrapping a fine-tuned gpt-3.5-turbo instance trained on prior ported features. This allows the model to innately take into account LangChain-specific code style and imports.It's an ongoing project, and feedback on runs will be used to improve the LangSmith dataset for further fine-tuning! Try it out below:https://langchain-translator.vercel.app/ Contributing Guidelines GitHub IssuesOur issues page contains

with bugs, improvements, and feature requests.If you start working on an issue, please assign it to yourself.If you are adding an issue, please try to keep it focused on a single modular bug/improvement/feature.

If the two issues are related, or blocking, please link them rather than keep them as one single one.We will try to keep these issues as up to date as possible, though

with the rapid rate of develop in this field some may get out of date.

If you notice this happening, please just let us know. Getting HelpAlthough we try to have a developer setup to make it as easy as possible for others to contribute (see below)

it is possible that some pain point may arise around environment setup, linting, documentation,

or other.

Should that occur, please contact a maintainer! Not only do we want to help get you unblocked, but we also want to make sure that the process is smooth for future contributors.In a similar vein, we do enforce certain linting, formatting, and documentation standards in the codebase.

If you are finding these difficult (or even just annoying) to work with,

feel free to contact a maintainer for help - we do not want these to get in the way of getting good code into the codebase. Release processAs of now, LangChain has an ad hoc release process: releases are cut with high frequency via by

a developer and published to npm.LangChain follows the semver versioning standard. However, as pre-1.0 software,

even patch releases may contain non-backwards-compatible changes.If your contribution has made its way into a release, we will want to give you credit on Twitter (only if you want though)!

If you have a Twitter account you would like us to mention, please let us know in the PR or in another manner. ToolingThis project uses the following tools, which are worth getting familiar

with if you plan to contribute:yarn (v3.4.1) - dependency managementeslint - enforcing standard lint rulesprettier - enforcing standard code formattingjest - testing codeTypeDoc - reference doc generation from

commentsDocusaurus - static site generation for documentation Quick StartClone this repo, then cd into it:cd langchainjsNext, try running the following common tasks: Common TasksOur goal is to make it as easy as possible for you to contribute to this project.

All of the below commands should be run from within the langchain/ directory unless otherwise noted.cd langchainSetupTo get started, you will need to install the dependencies for the project. To do so, run:yarnLintingWe use eslint to enforce standard lint rules.

To run the linter, run:yarn lintFormattingWe use prettier to enforce code formatting style.

To run the formatter, run:yarn formatTo just check for formatting differences, without fixing them, run:yarn format:checkTestingIn general, tests should be added within a tests/ folder

alongside the modules they

are testing.Unit tests cover modular logic that does not require calls to outside APIs.If you add new logic, please add a unit test.

Unit tests should be called *.test.ts.To run only unit tests, run:yarn testRunning a single testTo run a single test, run:yarn test:single /path/to/yourtest.test.tsThis is useful for developing individual features.Integration tests cover logic that requires making calls to outside APIs (often integration with other services).If you add support for a new external API, please add a new integration test.

Integration tests should be called *.int.test.ts.Note that most integration tests require credentials or other setup. You will likely need to set up a langchain/.env file

like the example here.We generally recommend only running integration tests with yarn test:single, but if you want to run all integration tests, run:yarn test:integrationBuildingTo build the project, run:yarn buildAdding an EntrypointLangChain exposes multiple subpaths the user can import from, e.g.import { OpenAI } from "langchain/llms/openai";We call these subpaths "entrypoints". In general, you should create a new entrypoint if you are adding a new integration with a 3rd party library. If you're adding self-contained functionality without any external dependencies, you can add it to an existing entrypoint.In order to declare a new entrypoint that users can import from, you

should edit the langchain/scripts/create-entrypoints.js script. To add an

entrypoint tools that imports from tools/index.ts you'd add

the following to the entrypoints variable:const entrypoints = { // ... tools: "tools/index",};This will make sure the entrypoint is included in the published package,

and in generated documentation.DocumentationContribute DocumentationDocs are largely autogenerated by TypeDoc from the code.For that reason, we ask that you add good documentation to all classes and methods.Similar to linting, we recognize documentation can be annoying. If you do not want to do it, please contact a project maintainer, and they can help you

with it. We do not want this to be a blocker for good code getting contributed.Documentation and the skeleton lives under the docs/ folder. Example code is imported from under the examples/ folder.Running examplesIf you add a new major piece of functionality, it is helpful to add an example to showcase how to use it. Most of our users find examples to be the most helpful kind of documentation.Examples can be added in the examples/src directory, e.g. examples/src/path/to/example and should export a run function. This example can then be invoked with yarn example path/to/example at the top level of the repo.To run examples that require an environment variable, you'll need to add a .env file under examples/.envBuild Documentation LocallyTo generate and view the documentation locally, change to the project root and run yarn to ensure dependencies get installed in both the docs/ and examples/ workspaces:cd ..yarnThen run:yarn docsAdvancedEnvironment tests test whether LangChain works across different JS environments, including Node.js (both ESM and CJS), Edge environments (eg. Cloudflare Workers), and browsers (using Webpack).To run the environment tests with Docker, run the following command from the project root:yarn test:exports:dockerCommunityDiscordTwitterGitHubPythonJS/TSMoreHomepageBlogCopyright © 2023 LangChain, Inc.