



【RooCode】4o に詳しく聞いてみたよ



れん

2025年8月5日 21:25

...

RooCodeをもっと使いこなしたいので、4o にしらべてもらったよ

わたしはVSCodeでRooCodeを利用しています
RooCodeで何ができるのかを知りたい
最新バージョンの情報も含めて教えて
<https://github.com/RooCodeInc/Roo-Code/blob/main/README.md>

RooCodeは Python、JSで主に使います。
Web系のアプリを制作します。
バックエンドはその時々で設計します。

今回はルミナ不在で調査してみました

了解しました。PythonとJavaScriptでのWeb系サービス開発において、VSCodeでRooCodeを使う際に何ができるのかを、レガシー機能も含めて網羅的に調査します。安定版の最新情報に限定し、開発中の機能は除外します。

調査が完了次第、要点をわかりやすく整理してご報告しますね。

RooCodeの概要と基本機能

RooCode（ルーコード）は、Visual Studio Code向けのAIコーディング支援拡張機能です。オープンソースで開発されており、元々は「Cline」というプロジェクトからフォークされて誕生しました。VSCode上に専用パネル（チャットUI）を追加し、**自然言語による指示**だけで様々なコーディングタスクを実行できます。具体的には、新しいコードの自動生成や既存コードの修正、コードに関する質問への回答、ドキュメントの作成更新など、**開発作業全般をAIエージェントが支援**します。PythonやJavaScriptを含む複数の言語・フレームワークに対応しており、特にWebサービス開発においてコーディングからデバッグまで幅広く活用できます。

主な特徴として、RooCodeは**「エディタ内のAI開発チーム」**とも称され、以下のような能力を持ちます：

- **自然な対話形式でのコーディング:** チャットに日本語や英語で「〇〇な機能を実装して」と指示するだけで、必要なコードを生成したり、コードレビュー・リファクタリング提案を行ったりします。複数ファイルにまたがる変更やプロジェクト全体の構成にも対応しており、**一度の指示で関連する複数のファイルを編集**できます。
- **開発環境への深い統合:** AIは**VSCode上のファイル群**を直接読み書きできるほか、**ターミナルコマンドの実行やブラウザ操作**まで自動化できます。たとえば、テストスクリプトを実行して結果を解析したり、依存パッケージのインストールコマンドを走らせたり、Webからドキュメントを取得して参照するといったことが可能です。
- **マルチエージェント（モード）機構:** 用途に応じてAIの****「役割」や「人格」を切り替えることができ、コーディング作業だけでなく設計レビューやデバッグ相談など複数の専門家チーム**がいるような使い分け**ができます。詳細は後述しますが、RooCodeには標準で5種類のモードが用意されており、それぞれAIの振る舞いと権限が異なります。

以上のように、RooCodeはVSCode上で**コード作成から実行・検証まで一貫してAIがサポート**してくれる強力なツールです。以下では、具体的な機能をカテゴリ別に整理し、Python/JS開発で有用なポイントやレガシー機能の現状、最新バージョン情報について詳述します。

VSCode拡張としての特長と統合機能

チャットベースのコード生成・編集機能

RooCodeの根幹は、**チャットUIでの対話によるコード生成・編集**です。エディタ左側のアクティビティバーに追加されるRooCodeアイコンをクリックすると、チャットペインが表示されます。ここでユーザは「自然言語の指示」を入力し、送信することでタスクを開始できます。例えば「dataフォルダ内にhello.pyを作成して、Helloと表示するPythonコードを書いて」や「この関数にバグがあるので修正して」といった自由なリクエストが可能です。**特別なコマンドや構文は不要で、日常言語でOK**です。

コード生成の流れ: ユーザがタスクを送信すると、RooCodeはまず**実行計画とアクション提案**を行います。例えば「ファイル作成」アクションでは作成されるファイル名と内容の差分、コード修正の場合は**差分(diff)形式**で追加・削除される行を表示し、**コマンド実行**の場合は実行するシェルコマンド、など具体的な操作内容をチャット上に提示します。これに対しユーザは**各アクションを承認(Approve)または拒否**できます。**承認すると実際にファイルに書き込みやコマンド実行が行われ、結果がまたフィードバックされます。**このようにユーザの確認を挟みながら**ステップごとに処理を進める**ため、誤った変更がいきなり適用される心配がなく安心です（自動承認を有効化することで確認プロンプトを省略することもできます）。最終的にタスクが完了すると、完了した旨と結果がチャットに表示されます。

マルチファイル対応とコンテキスト共有: RooCodeは**複数ファイルを跨いだ実装**に対応しており、プロジェクト全体を理解した上でコードを生成・修正します。個々のファイル内容をAIに正確に伝えるために、**「@」記法によるコンテキスト指定**が可能です。例えば「@utils.jsの関数の中身を教えて」とメッセージに書けば、そのファイル内容がAIに読み込まれます。@problemsと書けばVSCodeの**エラー一覧**（Problemsパネルの内容）を参照させることができます。@git-changesで未コミットの差分、@commit:HEAD~1で直前コミットの内容、@terminalで最後に実行したターミナル出力、@https://〜で指定URLのWebページ内容、といった具合に、**開発中によく使う情報をコピペせず共有できる**のが便利です。この機能により、たとえば「最近のコミットで変更した部分を見ながらテストコードを書いて」等と指示すれば、AIがそのコミット差分を理解した上でテストを生成してくれます。

ドキュメント生成・質問応答: コードそのものだけでなく、**ドキュメントの自動生成**やコードベースに関する質問への回答も行えます。例えば「この関数の処理をコメントで説明して」や「このプロジェクトでReactの状態管理はどう実装している？」と聞けば、該当コードを読み解いて**説明文や設計意図を回答**します。Python/JSのライブラリ使用方法やエラーメッセージの意味など**一般的なプログラミングQ&A**にも対応しており（Askモード利用時、後述）、スタックオーバーフローやドキュメントを調べ代わりにRooCodeに尋ねることもできます。

複数のモードによるAIエージェントの使い分け

RooCode最大の特徴の一つが、**目的別に最適化された複数のエージェント“モード”**を備えている点です。モードごとにAIの得意分野やツール使用権限が異なり、**仮想的に役割の異なる開発メンバー**を切り替えるような感覚で使い分けができます。最新安定版では、以下の5つのモードが標準搭載されています:

- **Codeモード（デフォルト）**: **プログラマー担当**のエージェントです。一般的なコーディングタスク全般（実装・リファクタリング・バグ修正など）に対応します。あらゆる開発ツールへのフルアクセス権限を持ち、自由にファイル編集やコマンド実行を行います。
- **Debugモード**: **不具合解析担当**のエージェントです。エラーログの分析、バグの原因特定、修正案の提示など、デバッグに特化したやり取りを行います。Codeモード同様に全ツールへのアクセス権限があり、問題箇所にログ出力を挿入したり、エラー再現のためコードを一部改変・実行したりといったアクションも取ります。内部的には原因仮説の列挙や再現確認→修正という**系統立ったアプローチ**を行うようカスタム指示が組まれており、複雑な不具合の調査に力を発揮します。
- **Askモード**: **技術アドバイザー担当**のエージェントです。コーディングを伴わない**純粋なQ&Aや調査**に向いており、例えば「このライブラリの具体的な使い方を教えて」や「非同期処理のしくみを図解して」等に対し、有用な情報を提供します。このモードでは**ファイル編集やコマンド実行は行わず**（閲覧系ツールのみ使用）、コードを書いて欲しい場合も明示的に指示しない限り解説に留めるなど、安全で丁寧な回答志向になります。
- **Architectモード**: **システムアーキテクト（設計担当）**のエージェントです。**技術選定やアーキテクチャ設計の相談、実装方針の立案など高レベルな議論**を行います。**ファイル編集は基本的に行わず**（テキストベースの設計ドキュメント程度に限定）、代わりに**構成図や設計プランの提案**を行う傾向があります。複数の選択肢を比較検討したり、ステップごとに要件整理から詳細設計まで段階的に**プランを練る**よう調整されています。
- **Orchestratorモード（別名: Boomerangモード）**: **プロジェクトマネージャ/リーダー担当**のエージェントです。他のモードを**オーケストレーション（統括制御）**する特殊なモードで、**複雑なタスクを自動で細分化し順次実行することを目的としています**。自身は**直接ファイル編集やコマンド実行をせず、必要に応じてnew_taskツールで他のモードにサブタスクを委譲する**形で動きます。例えば「この新機能の要件定義からテストまで一通りやって」と指示すると、OrchestratorモードがArchitect→Code→Debugと各モードを順次呼び出しつつ、要件を設計・実装・テストまで自動進行しようと試みます。プロジェクト開始から完了までを一括で**AIに任せる実験的フロー**として、上手くハマれば非常に強力なモードです。

モード切り替え方法: チャット入力欄の左にあるドロップダウンから選択するか、メッセージ先頭に/codeや/askなどスラッシュコマンドを打つことで即時にモードを変更できます。またキーボードショートカット（WindowsならCtrl + .）でモードを順に切り替えることも可能です。**モードごとに直前に使用したAIモデルを記憶**する仕組みもあり、たとえばCodeモードでは高速なコード生成向けモデル、Architectモードでは長文思考向けモデル、と使い分けておけばモード変更に応じ自動でモデルも切り替わります。このようにRooCodeは**状況に応じ最適なエージェント（+モデル）**で対話できる**柔軟性**があり、開発ワークフロー全体をシームレスにサポートします。

開発ツール統合: ターミナル・ブラウザ・ファイル操作

RooCodeはVSCode内の各種リソースや外部ツールと連携し、**AIに開発環境を直接操作させる**ことができます。主要な統合ポイントは次の通りです:

- **ファイルの読み書き**: AIはワークスペース内の任意のファイルを開いて内容を読み込んだり、新規ファイルの作成・編集・削除を行えます。例えば「README.mdにUsageセクションを追加して」と頼めば、そのファイルを読み既存内容を保持しつつ追記するといった作業をします。編集内容は**差分表示**され、ユーザが確認して承認してから実施されます。一度に複数ファイルを変更するような指示も可能で、その場合各ファイルごとに差分が提示され順次承認していきます。
- **ターミナル統合**: VSCodeの統合ターミナルと接続し、AIが**シェルコマンドを実行**できます。ビルドやテストの実行、依存関係のインストール、lintやフォーマッタの実行、さらにはgitコマンドでコミット/プッシュすることまで自動化可能です（※破壊的な操作は注意）。実行にはVSCodeのターミナル機能を内部利用しており、**リアルタイムで出力を取得して解析**できます。例えば「サーバーを起動して」と指示すればnpm run devを実行し、そのログを読んで**起動エラーが出たら自動で原因を特定・修正**するといったインテリジェントな振る舞いも可能です。「コマンド実行中は出力を逐次モニタし、必要なら停止する」という制御も利き、チャットUI上から**コマンド強制終了**ボタンを押すこともできます。このターミナル連携はRooCodeインストール時に自動有効化され特別な設定は不要ですが、正常に機能しない場合のフォールバック実行（バックグラウンドプロセスでの代

替実行）や対応Shellの指定、出力の最大取得行数調整など細かなオプションも用意されています。

- **デバッガー支援:** RooCode自体がVSCodeのデバッガー（ブレークポイントやステップ実行）を直接操作する機能はありませんが、その代替として**実行結果やエラー情報の収集・解析**を行います。上述のターミナル実行によりアプリケーションを起動し、そのログやスタックトレースをチェックして**原因箇所の候補を提示→該当コードの修正提案**まで自動で行います（Debugモード時に顕著）。さらにVSCodeの**問題パネル（Diagnostics）**とも連携しており、ビルドエラーやLint警告の一覧を@指定で読み込んで不具合修正に役立てられます。大量のエラーがある場合にどこまでコンテキストに含めるかも調整可能（Diagnostic設定で最大件数を制限する等）で、ノイズを減らしつつ重要な問題にフォーカスできます。
- **ブラウザ操作:** 組み込みの**ブラウザ操作ツール**を備えており、AIがインターネットから情報収集することもできます。指定URLのページ内容を読み込む、検索クエリを実行して得られた結果を要約する、といったことが可能です。特に**Google Gemini**系モデルを用いる際には、このWebアクセス機能が強化されており、実際のWeb情報に基づいた**最新の正確な回答**を返すこともできます。例えば「最新のReactのContext APIのベストプラクティスを調べてコード例を書いて」という指示では、検索により最新記事を読んで内容を反映した回答・コード生成を行うことが期待できます。
※Webアクセスはプライバシーのため設定で無効化することもできます。
- **プラグイン（MCP）拡張:** RooCodeは**MCP（Model Context Protocol）**という拡張フレームワークに対応しており、**外部ツールやAPIとの連携機能を追加**できます。公式には「**無制限のカスタムツールを追加可能**」とされており、例えばデータベース接続用のMCPやJiraチケット連携MCPなど、コミュニティ主導で様々な拡張が作られています。これらはRooCode内のマーケットプレイスからワンクリックでインストール可能で（プロジェクト単位 or グローバルに導入可）、自作のモードやMCPを共有することもできます。MCPによって使用できるAIモデルも拡張され、AnthropicのClaudeやGoogleのPalm、ローカルのLLMなど**OpenAI以外のモデル**も統一的に扱えます。

コード補完やLintとの併用

コード補完(オートコンプリート)について、RooCodeはGitHub Copilotのようなインライン補完エンジンではありません。チャットを介してある程度まとまったコード（関数単位やファイル単位）を生成・編集するのが基本であり、リアルタイムに入力途中の行を予測して提案するといった機能は持っていません。したがって、必要に応じて**VSCode標準のIntelliSense補完や別途導入した補完ツール（Copilot等）と併用**することになります。ただし、RooCodeには**「Code Actions」機能があり、エディタ上で選択したテキストに対し右クリックメニューやショートカットからRooCodeの操作を呼び出すことも可能です（例：「この選択範囲のコードをリファクタして」といったアクション）。このように補完は従来通り人間が行いつつ、RooCodeは大きな改変や生成を担当する**という使い分けが現実的です。

Lint統合については、RooCode自身がLintエンジンを持つわけではありませんが、**既存のLint/型チェック結果をAIが活用**できる仕組みがあります。前述の@problemsでVSCodeのエラー・警告一覧を取り込み、その内容に基づいて自動修正や改善提案を行います。例えばESLintやFlake8で多数の警告が出ている場合、「Lintエラーをすべて直して」と頼めば、AIが問題一覧を読み取り**一括で修正コードを提案**してくれることもあります。設定画面で「何件のDiagnostics（問題）を送るか」を制限することもでき、大規模プロジェクトで数百件の警告があるような場合でも、まずは上位いくつかに絞って処理させるといった調整が可能です（2025年4月頃のアップデートでDiagnostics制御機能が追加されています）。なお、AIがコードを書く際にも、組み込みの**ルールファイル**（後述）やLintの一般知識を活用して、コーディング規約に沿った出力を心がける傾向があります。

プロジェクト管理・タスク支援機能

RooCodeはコード生成だけでなく、**プロジェクト全体の進行管理やタスク整理**を支援する機能も備えています。大規模開発を見据え、以下のような仕組みが提供されています：

- **チェックポイント (Checkpoints):** 開発途中の状態を記録しておける**スナップショット機能**です。裏側ではGitと連携しており、重要なタイミングで自動的にプロジェクトの状態（ファイル内容）をコミットのように保存します。ユーザが明示的にチェックポイントを作成することもでき、誤った方向に進んだと感じたら**過去のチェックポイントにロールバック**する、といったことがGUI上で簡単にできます。チェックポイント間の差分比較や、特定チェックポイント時点のファイル内容をAIに参照させることも可能で、安心してAIに大規模編集を任せる手助けとなっています。
- **タスク・ToDoリスト:** 複数の開発タスクを扱う場合、RooCodeは**タスクリストUI**を提供します。チャットで「次に何をすべきか？」といったやり取りをするとAIがサブタスクを列挙してくれるこ

とありますが、これらを**Todoリスト**として管理し、完了チェックを付けたり順序を入れ替えたりできます。特にOrchestratorモードでは複雑なプロジェクトを自動化する際、内部でサブタスク群を扱いますが、その進行をユーザが視覚的に追えるようになっています（Task Todo List機能）。

- **プロジェクトテンプレート**: (一般的な意味でのプロジェクト雛形機能) RooCode自体がGUI上で特定の雛形プロジェクトを作成してくれる機能はありません。ただし、AIに「React + Expressのプロジェクトをひな形から作って」と依頼すれば、必要なファイル一式（package.jsonや設定ファイル、フォルダ構成など）を**ゼロから自動生成**させることができます。これに関してはOrchestratorモードが威力を発揮し、要件を伝えたと適切なテンプレートのコードを複数ファイルにわたり作成してくれるでしょう。つまり**固定のテンプレート集が内蔵されているわけではない**ものの、AIの知識によって自由度の高いプロジェクト雛形を構築できます。必要に応じてコミュニティ提供の**カスタムモード**（マーケットプレイス経由）で「Railsプロジェクト初期化モード」等を追加し、特定技術スタック向けテンプレートを即座に生成することも可能です。
- **コラボレーション機能**: 現時点（2025年8月）で、RooCodeに**リアルタイム共同編集**やマルチユーザコラボレーションを直接支援する機能はありません。他の開発者とAIのやり取り内容を共有したい場合は、チャットログをエクスポートして渡すなどの手動対応になります。ただし、チームでRooCodeを導入する際には**プロジェクト内に設定ファイルを含めて共有**することである程度環境を統一できます（後述の.rooフォルダ配下ルールや.roomodesファイルをリポジトリに含める等）。また、将来的な展望として**Roo Code Cloud**というクラウドサービスが予告されており、設定同期や共有MCPサーバなどが提供予定です（v3.23でクラウド待機リストへのサインアップ機能が追加されました）。しかしクラウド機能はまだ一般提供されていないため、現状でのチーム開発支援は**設定・ルール類の共有**に留まります。

カスタマイズとレガシー機能

RooCodeは高機能ですが、開発現場に合わせた**カスタマイズ性**も重視されています。加えて、初期バージョンから存在する**レガシー機能**も改良を重ねつつ今なお重要な役割を果たしています。ここではその代表例を紹介します。

- **ルールファイルによるガイドライン適用**: プロジェクト独自のコーディング規約や開発手順をAIに守らせるため、**.roo/rules-<モード名>.md**という**ルール定義ファイル**を置くことができます。例えば**Codeモード**用に**.roo/rules-code.md**を作成し、「**変数名はスネークケース**」「**テストコードも必ず同時に生成すること**」等**チームの決まり事**を書いておけば、**タスク実行時にAIがその内容を読み込み開発文化に沿った振る舞い**をしてくれます。この仕組みは初期から存在するRooCodeの特徴で、最新バージョンでも引き続き**チームの知見をAIに継承させる手段**として有効活用できます。
- **Custom Instructions（カスタム指示）**: ルールファイルと似ていますが、こちらは**ユーザ個人のデフォルト指示**を設定画面から与える機能です。いわゆるシステムプロンプトを編集できるもので、「出力は常に日本語で」や「コードブロックには言語名を付与して」など、自分好みの応答スタイルを予め仕込んでおけます。これも初期からある機能で、現在もUI上で容易に編集できるよう整備されています。
- **.rooignore**: プロジェクト内の一部ファイル（大量の依存ライブラリやビルド成果物など）を読み込み対象から除外するための設定ファイルです。gitignoreと同様の記法で指定でき、巨大ファイルを誤ってAIに読ませてトークン浪費するのを防ぎます。これも以前からある機能で、最新でも大規模プロジェクトでのパフォーマンスチューニングに欠かせません。
- **マルチモデル対応**: 当初RooCode（旧称RooCline）はOpenAI APIキーによるGPTモデル利用が中心でしたが、現在ではAnthropic ClaudeやGoogle PaLM2、Meta Llama2、さらにはLoRA適用したローカルLLMまで**多数のモデルプロバイダを切替可能**です。設定画面の「プロバイダ」から利用したいAPIを選びキーを登録することで、**複数プロバイダのプロファイルを使い分け**られます。例えば「高精度なコード生成はGPT-4、設計議論はClaude 2を使う」といった切替がワンクリックでできます。これらの機能はバージョンアップで徐々に拡充されてきたもので、**最新安定版では主要AIサービスのほぼ全てに対応**すると言ってよい状況です（v3.25ではCerebrasなど新プロバイダも追加されています）。
- **モード/ツールの追加拡張**: 先述のマーケットプレイス経由で**ユーザ定義のモード**やツール(MCP)をインストールできる機能も、コミュニティの発展とともに成熟してきた部分です。レガシー機能というより新機能寄りですが、当初は手動でyamlファイルを弄っていたカスタムモードも、現在ではUI上で検索・ワンクリック導入できるようになり利便性が向上しました。
- **Chatモード導入と進化**: 振り返ると**モード制自体**が途中から追加された機能です。2025年2月リリースのv3.0で初めてAskやArchitectといったチャットモードの概念が導入されるなど、RooCodeは黎明期から大きく進化してきました。初期には1種類だったAIエージェントが、今では5モード+カスタム無制限となり、これらレガシー機能の延長線上に最新機能が成り立っています。

以上のように、RooCodeは初期からの有用な機能を残しつつ着実に拡張・改善されています。特にマルチファイル編集・ターミナル実行・ルール適用といった基本機能は最初期から現在まで一貫して提供されるRooCodeの核であり、最新バージョンでも安定して利用できます。

最新安定版のバージョン情報とリリースノート

最新の安定バージョンは **RooCode v3.25系** です（執筆時点での最新版は **v3.25.6**）。バージョン3.25.0が2025年7月29日にリリースされ、その後も不具合修正や小機能追加のバッチが継続的にリリースされています。最新パッチv3.25.6は2025年8月1日に公開されました（3.25系としては6つ目の安定リリース）。

RooCode 3.25では、開発効率をさらに高めるための**新機能や改良**が多数導入されています。主なハイライトを挙げると：

- **メッセージのキューイング**: 処理中でも次々と指示を送信キューに積むことができるようになりました。これにより、AIの応答待ちで手が止まることなく、**連続して質問や追加指示を送れる**ようになっています。送信済みメッセージは順番に実行され、UI上で保留中のキューが表示されます。
- **カスタムスラッシュコマンド**: 独自のスラッシュコマンド（/deployや/reviewなど）をユーザが作成できるようになりました。プロジェクト内の.roo/commands/フォルダにMarkdownで定型プロンプトを記述すると、即座に**カスタムコマンドとして登録**されます。専用の管理UIも提供され、コマンドの追加・編集・削除が容易です。引数ヒントや説明文も付けられるため、チーム内で共有するWorkflowを**簡単なコマンド一つで再現**できます。
- **Geminiモデル向けWebアクセス強化**: GoogleのGeminiモデル使用時に、**指定URLのページ内容やGoogle検索結果**をリアルタイム取得して回答に反映できるようになりました。最新ドキュメントの参照や情報の事実確認を自動で行うことで、**回答精度の向上**が期待できます。プライバシー設定でこの機能のON/OFF切替も可能です。
- **画像ファイルの解析対応**: read_fileツールで**画像（PNG/JPG/GIF等）の読み取りとOCR解析**がサポートされました。スクリーンショット内のテキスト抽出やUIモック画像からの情報読み取りが可能です。複数画像をまとめて読み込むこともでき、画像からコードやデザインを読み取って活用できる点でユニークな拡張です。
- **コスト制御機能の強化**: AI API利用コストがかさみすぎないように、**自動承認時のコスト上限**を設定する機能が追加されました。一定金額/トークン数を超える操作は自動実行しないようにでき、予期せぬ課金を防ぎます。
- **プロバイダ対応の拡充**: 新たにCerebras AIプロバイダが公式サポートされ、同社のQwen-3 Coderモデルなどが利用可能になりました。他にもOpenRouter経由でのhorizonシリーズモデル対応や、中国企業の大規模モデル（DoubaoやGLM4.5など）の追加統合など、**利用できるモデルの選択肢がさらに広がっています。

この他、**UIの改良**（マーケットプレイス情報の自動更新やボタン挙動改善など）や**不具合修正**（コマンド実行の安定性向上、Lint対象漏れの修正等）も多数含まれています。詳しくは公式のリリースノート（CHANGELOG）やドキュメントのUpdate Notesセクションにバージョンごとの変更点一覧が公開されています。

以上、RooCodeの最新安定版の機能について、**Python/JavaScript開発に関連するポイントを中心に包括的に**まとめました。VSCodeユーザーにとってRooCodeは、単なるコード補完に留まらない**強力な開発パートナー**となり得る拡張機能です。コード自動生成やリファクタはもちろん、ターミナルやブラウザまで巻き込んだ自動化、モードによる柔軟な支援、そしてチーム規約の適用など、多彩な機能を備えています。最新バージョンでは使い勝手と機能性がさらに向上しており、**現場への導入メリットは非常に大きい**でしょう。ぜひプロジェクトに合わせた設定を行い、RooCodeを活用して効率的なPython/JS開発を実現してみてください。

何回か読み直しが必要だな