# TypeScript ESTree

A parser that converts TypeScript source code into an [ESTree](https://github.com/estree/estree)-compatible form

    

## About This parser is somewhat generic and robust, and could be used to power any use-case which requires taking TypeScript source code and producing an ESTree-compatible AST. In fact, it is already used within these hyper-popular open-source projects to power their TypeScript support: - [ESLint](https://eslint.org), the pluggable linting utility for JavaScript and JSX - [Prettier](https://prettier.io), an opinionated code formatter ## Installation ```sh yarn add -D @typescript-eslint/typescript-estree ``` ## API ### Parsing #### `parse(code, options)` Parses the given string of code with the options provided and returns an ESTree-compatible AST. ```ts interface ParseOptions { /\*\* \* create a top-level comments array containing all comments \*/ comment?: boolean; /\*\* \* An array of modules to turn explicit debugging on for. \* - 'typescript-eslint' is the same as setting the env var `DEBUG=typescript-eslint:\*` \* - 'eslint' is the same as setting the env var `DEBUG=eslint:\*` \* - 'typescript' is the same as setting `extendedDiagnostics: true` in your tsconfig compilerOptions \* \* For convenience, also supports a boolean: \* - true === ['typescript-eslint'] \* - false === [] \*/ debugLevel?: boolean | ('typescript-eslint' | 'eslint' | 'typescript')[]; /\*\* \* Cause the parser to error if it encounters an unknown AST node type (useful for testing). \* This case only usually occurs when TypeScript releases new features. \*/ errorOnUnknownASTType?: boolean; /\*\* \* Absolute (or relative to `cwd`) path to the file being parsed. \*/ filePath?: string; /\*\* \* Enable parsing of JSX. \* For more details, see https://www.typescriptlang.org/docs/handbook/jsx.html \* \* NOTE: this setting does not effect known file types (.js, .jsx, .ts, .tsx, .json) because the \* TypeScript compiler has its own internal handling for known file extensions. \* \* For the exact behavior, see https://github.com/typescript-eslint/typescript-eslint/tree/master/packages/parser#parseroptionsecmafeaturesjsx \*/ jsx?: boolean; /\*\* \* Controls whether the `loc` information to each node. \* The `loc` property is an object which contains the exact line/column the node starts/ends on. \* This is similar to the `range` property, except it is line/column relative. \*/ loc?: boolean; /\* \* Allows overriding of function used for logging. \* When value is `false`, no logging will occur. \* When value is not provided, `console.log()` will be used. \*/ loggerFn?: Function | false; /\*\* \* Controls whether the `range` property is included on AST nodes. \* The `range` property is a [number, number] which indicates the start/end index of the node in the file contents. \* This is similar to the `loc` property, except this is the absolute index. \*/ range?: boolean; /\*\* \* Set to true to create a top-level array containing all tokens from the file. \*/ tokens?: boolean; /\* \* The JSX AST changed the node type for string literals \* inside a JSX Element from `Literal` to `JSXText`. \* When value is `true`, these nodes will be parsed as type `JSXText`. \* When value is `false`, these nodes will be parsed as type `Literal`. \*/ useJSXTextNode?: boolean; } const PARSE\_DEFAULT\_OPTIONS: ParseOptions = { comment: false, errorOnUnknownASTType: false, filePath: 'estree.ts', // or 'estree.tsx', if you pass jsx: true jsx: false, loc: false, loggerFn: undefined, range: false, tokens: false, useJSXTextNode: false, }; declare function parse( code: string, options: ParseOptions = PARSE\_DEFAULT\_OPTIONS, ): TSESTree.Program; ``` Example usage: ```js import { parse } from '@typescript-eslint/typescript-estree'; const code = `const hello: string = 'world';`; const ast = parse(code, { loc: true, range: true, }); ``` #### `parseAndGenerateServices(code, options)` Parses the given string of code with the options provided and returns an ESTree-compatible AST. Accepts additional options which can be used to generate type information along with the AST. ```ts interface ParseAndGenerateServicesOptions extends ParseOptions { /\*\* \* Causes the parser to error if the TypeScript compiler returns any unexpected syntax/semantic errors. \*/ errorOnTypeScriptSyntacticAndSemanticIssues?: boolean; /\*\* \* When `project` is provided, this controls the non-standard file extensions which will be parsed. \* It accepts an array of file extensions, each preceded by a `.`. \*/ extraFileExtensions?: string[]; /\*\* \* Absolute (or relative to `tsconfigRootDir`) path to the file being parsed. \* When `project` is provided, this is required, as it is used to fetch the file from the TypeScript compiler's cache. \*/ filePath?: string; /\*\* \* Allows the user to control whether or not two-way AST node maps are preserved \* during the AST conversion process. \* \* By default: the AST node maps are NOT preserved, unless `project` has been specified, \* in which case the maps are made available on the returned `parserServices`. \* \* NOTE: If `preserveNodeMaps` is explicitly set by the user, it will be respected, \* regardless of whether or not `project` is in use. \*/ preserveNodeMaps?: boolean; /\*\* \* Absolute (or relative to `tsconfigRootDir`) paths to the tsconfig(s). \* If this is provided, type information will be returned. \*/ project?: string | string[]; /\*\* \* If you provide a glob (or globs) to the project option, you can use this option to blacklist \* certain folders from being matched by the globs. \* Any project path that matches one or more of the provided regular expressions will be removed from the list. \* \* Accepts an array of strings that are passed to new RegExp(), or an array of regular expressions. \* By default, this is set to ["/node\_modules/"] \*/ projectFolderIgnoreList?: (string | RegExp)[]; /\*\* \* The absolute path to the root directory for all provided `project`s. \*/ tsconfigRootDir?: string; /\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \* IT IS RECOMMENDED THAT YOU DO NOT USE THIS OPTION, AS IT CAUSES PERFORMANCE ISSUES. \* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \* \* When passed with `project`, this allows the parser to create a catch-all, default program. \* This means that if the parser encounters a file not included in any of the provided `project`s, \* it will not error, but will instead parse the file and its dependencies in a new program. \*/ createDefaultProgram?: boolean; } const PARSE\_AND\_GENERATE\_SERVICES\_DEFAULT\_OPTIONS: ParseOptions = { ...PARSE\_DEFAULT\_OPTIONS, errorOnTypeScriptSyntacticAndSemanticIssues: false, extraFileExtensions: [], preserveNodeMaps: false, // or true, if you do not set this, but pass `project` project: undefined, projectFolderIgnoreList: ['/node\_modules/'], tsconfigRootDir: process.cwd(), }; declare function parseAndGenerateServices( code: string, options: ParseOptions = PARSE\_DEFAULT\_OPTIONS, ): TSESTree.Program; ``` Example usage: ```js import { parseAndGenerateServices } from '@typescript-eslint/typescript-estree'; const code = `const hello: string = 'world';`; const ast = parseAndGenerateServices(code, { filePath: '/some/path/to/file/foo.ts', loc: true, project: './tsconfig.json', range: true, }); ``` ### `TSESTree`, `AST\_NODE\_TYPES` and `AST\_TOKEN\_TYPES` Types for the AST produced by the parse functions. - `TSESTree` is a namespace which contains object types representing all of the AST Nodes produced by the parser. - `AST\_NODE\_TYPES` is an enum which provides the values for every single AST node's `type` property. - `AST\_TOKEN\_TYPES` is an enum which provides the values for every single AST token's `type` property. ## Supported TypeScript Version We will always endeavor to support the latest stable version of TypeScript. The version of TypeScript currently supported by this parser is `~3.2.1`. This is reflected in the `devDependency` requirement within the package.json file, and it is what the tests will be run against. We have an open `peerDependency` requirement in order to allow for experimentation on newer/beta versions of TypeScript. If you use a non-supported version of TypeScript, the parser will log a warning to the console. \*\*Please ensure that you are using a supported version before submitting any issues/bug reports.\*\* ## Reporting Issues Please check the current list of open and known issues and ensure the issue has not been reported before. When creating a new issue provide as much information about your environment as possible. This includes: - TypeScript version - The `typescript-estree` version ## AST Alignment Tests A couple of years after work on this parser began, the TypeScript Team at Microsoft began [officially supporting TypeScript parsing via Babel](https://blogs.msdn.microsoft.com/typescript/2018/08/27/typescript-and-babel-7/). I work closely with the TypeScript Team and we are gradually aligning the AST of this project with the one produced by Babel's parser. To that end, I have created a full test harness to compare the ASTs of the two projects which runs on every PR, please see the code for more details. ## Build/Test Commands - `npm test` - run all tests - `npm run unit-tests` - run only unit tests - `npm run ast-alignment-tests` - run only Babylon AST alignment tests ## Debugging If you encounter a bug with the parser that you want to investigate, you can turn on the debug logging via setting the environment variable: `DEBUG=typescript-eslint:\*`. I.e. in this repo you can run: `DEBUG=typescript-eslint:\* yarn lint`. ## License TypeScript ESTree inherits from the the original TypeScript ESLint Parser license, as the majority of the work began there. It is licensed under a permissive BSD 2-clause license. ## Contributing [See the contributing guide here](../../CONTRIBUTING.md)