# ESLint Plugin TypeScript

    

## Getting Started - \*\*[You can find our Getting Started docs here](../../docs/getting-started/linting/README.md)\*\* - \*\*[You can find our FAQ / Troubleshooting docs here](../../docs/getting-started/linting/FAQ.md)\*\* These docs walk you through setting up ESLint, this plugin, and our parser. If you know what you're doing and just want to quick start, read on... ## Quick-start ### Installation Make sure you have TypeScript and [`@typescript-eslint/parser`](../parser) installed, then install the plugin: ```sh yarn add -D @typescript-eslint/eslint-plugin ``` It is important that you use the same version number for `@typescript-eslint/parser` and `@typescript-eslint/eslint-plugin`. \*\*Note:\*\* If you installed ESLint globally (using the `-g` flag) then you must also install `@typescript-eslint/eslint-plugin` globally. ### Usage Add `@typescript-eslint/parser` to the `parser` field and `@typescript-eslint` to the plugins section of your `.eslintrc` configuration file, then configure the rules you want to use under the rules section. ```json { "parser": "@typescript-eslint/parser", "plugins": ["@typescript-eslint"], "rules": { "@typescript-eslint/rule-name": "error" } } ``` You can also enable all the recommended rules for our plugin. Add `plugin:@typescript-eslint/recommended` in extends: ```json { "extends": ["plugin:@typescript-eslint/recommended"] } ``` \*\*Note: Make sure to use `eslint --ext .js,.ts` since by [default](https://eslint.org/docs/user-guide/command-line-interface#--ext) `eslint` will only search for `.js` files.\*\* ### Recommended Configs You can also use [`eslint:recommended`](https://eslint.org/docs/rules/) (the set of rules which are recommended for all projects by the ESLint Team) with this plugin. As noted in the root README, not all ESLint core rules are compatible with TypeScript, so you need to add both `eslint:recommended` and `plugin:@typescript-eslint/eslint-recommended` (which will adjust the one from ESLint appropriately for TypeScript) to your config: ```json { "extends": [ "eslint:recommended", "plugin:@typescript-eslint/eslint-recommended", "plugin:@typescript-eslint/recommended" ] } ``` As of version 2 of this plugin, \_by design\_, none of the rules in the main `recommended` config require type-checking in order to run. This means that they are more lightweight and faster to run. Some highly valuable rules simply require type-checking in order to be implemented correctly, however, so we provide an additional config you can extend from called `recommended-requiring-type-checking`. You would apply this \_in addition\_ to the recommended configs previously mentioned, e.g.: ```json { "extends": [ "eslint:recommended", "plugin:@typescript-eslint/eslint-recommended", "plugin:@typescript-eslint/recommended", "plugin:@typescript-eslint/recommended-requiring-type-checking" ] } ``` Pro Tip: For larger codebases you may want to consider splitting our linting into two separate stages: 1. fast feedback rules which operate purely based on syntax (no type-checking), 2. rules which are based on semantics (type-checking). \*\*[You can read more about linting with type information here](../../docs/getting-started/linting/TYPED\_LINTING.md)\*\* ## Supported Rules \*\*Key\*\*: :heavy\_check\_mark: = recommended, :wrench: = fixable, :thought\_balloon: = requires type information | Name | Description | :heavy\_check\_mark: | :wrench: | :thought\_balloon: | | --------------------------------------------------------------------------------------------------------------------- | ----------------------------------------------------------------------------------------------------------------------- | ------------------ | -------- | ----------------- | | [`@typescript-eslint/adjacent-overload-signatures`](./docs/rules/adjacent-overload-signatures.md) | Require that member overloads be consecutive | :heavy\_check\_mark: | | | | [`@typescript-eslint/array-type`](./docs/rules/array-type.md) | Requires using either `T[]` or `Array` for arrays | | :wrench: | | | [`@typescript-eslint/await-thenable`](./docs/rules/await-thenable.md) | Disallows awaiting a value that is not a Thenable | :heavy\_check\_mark: | | :thought\_balloon: | | [`@typescript-eslint/ban-ts-comment`](./docs/rules/ban-ts-comment.md) | Bans `// @ts-` comments from being used | | | | | [`@typescript-eslint/ban-types`](./docs/rules/ban-types.md) | Bans specific types from being used | :heavy\_check\_mark: | :wrench: | | | [`@typescript-eslint/class-literal-property-style`](./docs/rules/class-literal-property-style.md) | Ensures that literals on classes are exposed in a consistent style | | :wrench: | | | [`@typescript-eslint/consistent-type-assertions`](./docs/rules/consistent-type-assertions.md) | Enforces consistent usage of type assertions | :heavy\_check\_mark: | | | | [`@typescript-eslint/consistent-type-definitions`](./docs/rules/consistent-type-definitions.md) | Consistent with type definition either `interface` or `type` | | :wrench: | | | [`@typescript-eslint/explicit-function-return-type`](./docs/rules/explicit-function-return-type.md) | Require explicit return types on functions and class methods | :heavy\_check\_mark: | | | | [`@typescript-eslint/explicit-member-accessibility`](./docs/rules/explicit-member-accessibility.md) | Require explicit accessibility modifiers on class properties and methods | | :wrench: | | | [`@typescript-eslint/explicit-module-boundary-types`](./docs/rules/explicit-module-boundary-types.md) | Require explicit return and argument types on exported functions' and classes' public class methods | | | | | [`@typescript-eslint/member-delimiter-style`](./docs/rules/member-delimiter-style.md) | Require a specific member delimiter style for interfaces and type literals | :heavy\_check\_mark: | :wrench: | | | [`@typescript-eslint/member-ordering`](./docs/rules/member-ordering.md) | Require a consistent member declaration order | | | | | [`@typescript-eslint/method-signature-style`](./docs/rules/method-signature-style.md) | Enforces using a particular method signature syntax. | | :wrench: | | | [`@typescript-eslint/naming-convention`](./docs/rules/naming-convention.md) | Enforces naming conventions for everything across a codebase | | | :thought\_balloon: | | [`@typescript-eslint/no-base-to-string`](./docs/rules/no-base-to-string.md) | Requires that `.toString()` is only called on objects which provide useful information when stringified | | | :thought\_balloon: | | [`@typescript-eslint/no-dynamic-delete`](./docs/rules/no-dynamic-delete.md) | Disallow the delete operator with computed key expressions | | :wrench: | | | [`@typescript-eslint/no-empty-interface`](./docs/rules/no-empty-interface.md) | Disallow the declaration of empty interfaces | :heavy\_check\_mark: | :wrench: | | | [`@typescript-eslint/no-explicit-any`](./docs/rules/no-explicit-any.md) | Disallow usage of the `any` type | :heavy\_check\_mark: | :wrench: | | | [`@typescript-eslint/no-extra-non-null-assertion`](./docs/rules/no-extra-non-null-assertion.md) | Disallow extra non-null assertion | | :wrench: | | | [`@typescript-eslint/no-extraneous-class`](./docs/rules/no-extraneous-class.md) | Forbids the use of classes as namespaces | | | | | [`@typescript-eslint/no-floating-promises`](./docs/rules/no-floating-promises.md) | Requires Promise-like values to be handled appropriately | | | :thought\_balloon: | | [`@typescript-eslint/no-for-in-array`](./docs/rules/no-for-in-array.md) | Disallow iterating over an array with a for-in loop | :heavy\_check\_mark: | | :thought\_balloon: | | [`@typescript-eslint/no-implied-eval`](./docs/rules/no-implied-eval.md) | Disallow the use of `eval()`-like methods | | | :thought\_balloon: | | [`@typescript-eslint/no-inferrable-types`](./docs/rules/no-inferrable-types.md) | Disallows explicit type declarations for variables or parameters initialized to a number, string, or boolean | :heavy\_check\_mark: | :wrench: | | | [`@typescript-eslint/no-invalid-void-type`](./docs/rules/no-invalid-void-type.md) | Disallows usage of `void` type outside of generic or return types | | | | | [`@typescript-eslint/no-misused-new`](./docs/rules/no-misused-new.md) | Enforce valid definition of `new` and `constructor` | :heavy\_check\_mark: | | | | [`@typescript-eslint/no-misused-promises`](./docs/rules/no-misused-promises.md) | Avoid using promises in places not designed to handle them | :heavy\_check\_mark: | | :thought\_balloon: | | [`@typescript-eslint/no-namespace`](./docs/rules/no-namespace.md) | Disallow the use of custom TypeScript modules and namespaces | :heavy\_check\_mark: | | | | [`@typescript-eslint/no-non-null-asserted-optional-chain`](./docs/rules/no-non-null-asserted-optional-chain.md) | Disallows using a non-null assertion after an optional chain expression | | | | | [`@typescript-eslint/no-non-null-assertion`](./docs/rules/no-non-null-assertion.md) | Disallows non-null assertions using the `!` postfix operator | :heavy\_check\_mark: | | | | [`@typescript-eslint/no-parameter-properties`](./docs/rules/no-parameter-properties.md) | Disallow the use of parameter properties in class constructors | | | | | [`@typescript-eslint/no-require-imports`](./docs/rules/no-require-imports.md) | Disallows invocation of `require()` | | | | | [`@typescript-eslint/no-this-alias`](./docs/rules/no-this-alias.md) | Disallow aliasing `this` | :heavy\_check\_mark: | | | | [`@typescript-eslint/no-throw-literal`](./docs/rules/no-throw-literal.md) | Disallow throwing literals as exceptions | | | :thought\_balloon: | | [`@typescript-eslint/no-type-alias`](./docs/rules/no-type-alias.md) | Disallow the use of type aliases | | | | | [`@typescript-eslint/no-unnecessary-boolean-literal-compare`](./docs/rules/no-unnecessary-boolean-literal-compare.md) | Flags unnecessary equality comparisons against boolean literals | | :wrench: | :thought\_balloon: | | [`@typescript-eslint/no-unnecessary-condition`](./docs/rules/no-unnecessary-condition.md) | Prevents conditionals where the type is always truthy or always falsy | | :wrench: | :thought\_balloon: | | [`@typescript-eslint/no-unnecessary-qualifier`](./docs/rules/no-unnecessary-qualifier.md) | Warns when a namespace qualifier is unnecessary | | :wrench: | :thought\_balloon: | | [`@typescript-eslint/no-unnecessary-type-arguments`](./docs/rules/no-unnecessary-type-arguments.md) | Enforces that type arguments will not be used if not required | | :wrench: | :thought\_balloon: | | [`@typescript-eslint/no-unnecessary-type-assertion`](./docs/rules/no-unnecessary-type-assertion.md) | Warns if a type assertion does not change the type of an expression | :heavy\_check\_mark: | :wrench: | :thought\_balloon: | | [`@typescript-eslint/no-unsafe-assignment`](./docs/rules/no-unsafe-assignment.md) | Disallows assigning any to variables and properties | | | :thought\_balloon: | | [`@typescript-eslint/no-unsafe-call`](./docs/rules/no-unsafe-call.md) | Disallows calling an any type value | | | :thought\_balloon: | | [`@typescript-eslint/no-unsafe-member-access`](./docs/rules/no-unsafe-member-access.md) | Disallows member access on any typed variables | | | :thought\_balloon: | | [`@typescript-eslint/no-unsafe-return`](./docs/rules/no-unsafe-return.md) | Disallows returning any from a function | | | :thought\_balloon: | | [`@typescript-eslint/no-unused-vars-experimental`](./docs/rules/no-unused-vars-experimental.md) | Disallow unused variables and arguments | | | :thought\_balloon: | | [`@typescript-eslint/no-var-requires`](./docs/rules/no-var-requires.md) | Disallows the use of require statements except in import statements | :heavy\_check\_mark: | | | | [`@typescript-eslint/prefer-as-const`](./docs/rules/prefer-as-const.md) | Prefer usage of `as const` over literal type | | :wrench: | | | [`@typescript-eslint/prefer-for-of`](./docs/rules/prefer-for-of.md) | Prefer a ‘for-of’ loop over a standard ‘for’ loop if the index is only used to access the array being iterated | | | | | [`@typescript-eslint/prefer-function-type`](./docs/rules/prefer-function-type.md) | Use function types instead of interfaces with call signatures | | :wrench: | | | [`@typescript-eslint/prefer-includes`](./docs/rules/prefer-includes.md) | Enforce `includes` method over `indexOf` method | :heavy\_check\_mark: | :wrench: | :thought\_balloon: | | [`@typescript-eslint/prefer-namespace-keyword`](./docs/rules/prefer-namespace-keyword.md) | Require the use of the `namespace` keyword instead of the `module` keyword to declare custom TypeScript modules | :heavy\_check\_mark: | :wrench: | | | [`@typescript-eslint/prefer-nullish-coalescing`](./docs/rules/prefer-nullish-coalescing.md) | Enforce the usage of the nullish coalescing operator instead of logical chaining | | :wrench: | :thought\_balloon: | | [`@typescript-eslint/prefer-optional-chain`](./docs/rules/prefer-optional-chain.md) | Prefer using concise optional chain expressions instead of chained logical ands | | :wrench: | | | [`@typescript-eslint/prefer-readonly`](./docs/rules/prefer-readonly.md) | Requires that private members are marked as `readonly` if they're never modified outside of the constructor | | :wrench: | :thought\_balloon: | | [`@typescript-eslint/prefer-readonly-parameter-types`](./docs/rules/prefer-readonly-parameter-types.md) | Requires that function parameters are typed as readonly to prevent accidental mutation of inputs | | | :thought\_balloon: | | [`@typescript-eslint/prefer-reduce-type-parameter`](./docs/rules/prefer-reduce-type-parameter.md) | Prefer using type parameter when calling `Array#reduce` instead of casting | | :wrench: | :thought\_balloon: | | [`@typescript-eslint/prefer-regexp-exec`](./docs/rules/prefer-regexp-exec.md) | Enforce that `RegExp#exec` is used instead of `String#match` if no global flag is provided | :heavy\_check\_mark: | | :thought\_balloon: | | [`@typescript-eslint/prefer-string-starts-ends-with`](./docs/rules/prefer-string-starts-ends-with.md) | Enforce the use of `String#startsWith` and `String#endsWith` instead of other equivalent methods of checking substrings | :heavy\_check\_mark: | :wrench: | :thought\_balloon: | | [`@typescript-eslint/prefer-ts-expect-error`](./docs/rules/prefer-ts-expect-error.md) | Recommends using `// @ts-expect-error` over `// @ts-ignore` | | :wrench: | | | [`@typescript-eslint/promise-function-async`](./docs/rules/promise-function-async.md) | Requires any function or method that returns a Promise to be marked async | | | :thought\_balloon: | | [`@typescript-eslint/require-array-sort-compare`](./docs/rules/require-array-sort-compare.md) | Requires `Array#sort` calls to always provide a `compareFunction` | | | :thought\_balloon: | | [`@typescript-eslint/restrict-plus-operands`](./docs/rules/restrict-plus-operands.md) | When adding two variables, operands must both be of type number or of type string | | | :thought\_balloon: | | [`@typescript-eslint/restrict-template-expressions`](./docs/rules/restrict-template-expressions.md) | Enforce template literal expressions to be of string type | | | :thought\_balloon: | | [`@typescript-eslint/strict-boolean-expressions`](./docs/rules/strict-boolean-expressions.md) | Restricts the types allowed in boolean expressions | | | :thought\_balloon: | | [`@typescript-eslint/switch-exhaustiveness-check`](./docs/rules/switch-exhaustiveness-check.md) | Exhaustiveness checking in switch with union type | | | :thought\_balloon: | | [`@typescript-eslint/triple-slash-reference`](./docs/rules/triple-slash-reference.md) | Sets preference level for triple slash directives versus ES6-style import declarations | :heavy\_check\_mark: | | | | [`@typescript-eslint/type-annotation-spacing`](./docs/rules/type-annotation-spacing.md) | Require consistent spacing around type annotations | :heavy\_check\_mark: | :wrench: | | | [`@typescript-eslint/typedef`](./docs/rules/typedef.md) | Requires type annotations to exist | | | | | [`@typescript-eslint/unbound-method`](./docs/rules/unbound-method.md) | Enforces unbound methods are called with their expected scope | :heavy\_check\_mark: | | :thought\_balloon: | | [`@typescript-eslint/unified-signatures`](./docs/rules/unified-signatures.md) | Warns for any two overloads that could be unified into one by using a union or an optional/rest parameter | | | | ### Extension Rules In some cases, ESLint provides a rule itself, but it doesn't support TypeScript syntax; either it crashes, or it ignores the syntax, or it falsely reports against it. In these cases, we create what we call an extension rule; a rule within our plugin that has the same functionality, but also supports TypeScript. \*\*Key\*\*: :heavy\_check\_mark: = recommended, :wrench: = fixable, :thought\_balloon: = requires type information | Name | Description | :heavy\_check\_mark: | :wrench: | :thought\_balloon: | | ----------------------------------------------------------------------------------------------- | ------------------------------------------------------------------------------ | ------------------ | -------- | ----------------- | | [`@typescript-eslint/brace-style`](./docs/rules/brace-style.md) | Enforce consistent brace style for blocks | | :wrench: | | | [`@typescript-eslint/comma-spacing`](./docs/rules/comma-spacing.md) | Enforces consistent spacing before and after commas | | :wrench: | | | [`@typescript-eslint/default-param-last`](./docs/rules/default-param-last.md) | Enforce default parameters to be last | | | | | [`@typescript-eslint/dot-notation`](./docs/rules/dot-notation.md) | enforce dot notation whenever possible | | :wrench: | :thought\_balloon: | | [`@typescript-eslint/func-call-spacing`](./docs/rules/func-call-spacing.md) | Require or disallow spacing between function identifiers and their invocations | | :wrench: | | | [`@typescript-eslint/indent`](./docs/rules/indent.md) | Enforce consistent indentation | | :wrench: | | | [`@typescript-eslint/init-declarations`](./docs/rules/init-declarations.md) | require or disallow initialization in variable declarations | | | | | [`@typescript-eslint/keyword-spacing`](./docs/rules/keyword-spacing.md) | Enforce consistent spacing before and after keywords | | :wrench: | | | [`@typescript-eslint/lines-between-class-members`](./docs/rules/lines-between-class-members.md) | Require or disallow an empty line between class members | | :wrench: | | | [`@typescript-eslint/no-array-constructor`](./docs/rules/no-array-constructor.md) | Disallow generic `Array` constructors | :heavy\_check\_mark: | :wrench: | | | [`@typescript-eslint/no-dupe-class-members`](./docs/rules/no-dupe-class-members.md) | Disallow duplicate class members | | | | | [`@typescript-eslint/no-empty-function`](./docs/rules/no-empty-function.md) | Disallow empty functions | :heavy\_check\_mark: | | | | [`@typescript-eslint/no-extra-parens`](./docs/rules/no-extra-parens.md) | Disallow unnecessary parentheses | | :wrench: | | | [`@typescript-eslint/no-extra-semi`](./docs/rules/no-extra-semi.md) | Disallow unnecessary semicolons | | :wrench: | | | [`@typescript-eslint/no-invalid-this`](./docs/rules/no-invalid-this.md) | disallow `this` keywords outside of classes or class-like objects | | | | | [`@typescript-eslint/no-magic-numbers`](./docs/rules/no-magic-numbers.md) | Disallow magic numbers | | | | | [`@typescript-eslint/no-unused-expressions`](./docs/rules/no-unused-expressions.md) | Disallow unused expressions | | | | | [`@typescript-eslint/no-unused-vars`](./docs/rules/no-unused-vars.md) | Disallow unused variables | :heavy\_check\_mark: | | | | [`@typescript-eslint/no-use-before-define`](./docs/rules/no-use-before-define.md) | Disallow the use of variables before they are defined | :heavy\_check\_mark: | | | | [`@typescript-eslint/no-useless-constructor`](./docs/rules/no-useless-constructor.md) | Disallow unnecessary constructors | | | | | [`@typescript-eslint/quotes`](./docs/rules/quotes.md) | Enforce the consistent use of either backticks, double, or single quotes | | :wrench: | | | [`@typescript-eslint/require-await`](./docs/rules/require-await.md) | Disallow async functions which have no `await` expression | :heavy\_check\_mark: | | :thought\_balloon: | | [`@typescript-eslint/return-await`](./docs/rules/return-await.md) | Enforces consistent returning of awaited values | | :wrench: | :thought\_balloon: | | [`@typescript-eslint/semi`](./docs/rules/semi.md) | Require or disallow semicolons instead of ASI | | :wrench: | | | [`@typescript-eslint/space-before-function-paren`](./docs/rules/space-before-function-paren.md) | Enforces consistent spacing before function parenthesis | | :wrench: | | ## Contributing [See the contributing guide here](../../CONTRIBUTING.md).