Changelog

v5.46.0

- Add “observedAttributes” domprop (#1652)
- More domprops (mostly `Temporal` related) suggested in #1652

v5.45.0

- Produce `void 0` instead of `undefined`, which is more safe

v5.44.1

- fix bitwise optimization changing the result of &&, `||`
- switches: make sure `var` is extracted from a deleted default case

v5.44.0

- Support `using` and `await using` declarations (#1635)

v5.43.1

- Prevent niche optimizations that would move around block declarations
- Add `lhs_constants` to `CompressOptions` type (#1621)

v5.43.0

- Do not wrap callbacks in parentheses (`wrap_func_args` format option is now false by default)
- Do not inline functions into for loops (for performance reasons)

v5.42.0

- Improved performance in the parse step by adding a fast path for simple identifiers.

- Improved ESTree conversion

v5.41.0

- fixed semicolon insertion between class fields, when the field names are number literals
- `keep_numbers` format option now works for bigint
- internal: correctly mark accessors' is_generator property
- internal: do not read or assign quote properties without need
- internal: add missing equivalent_to comparison

v5.40.0

- Fix exporting AssignmentExpression (default assign pattern) to ESTree
- Fix ESTree output of object keys with quotes
- Fix handling of an ESTree empty `export {}` (#1601)
- Fix some `const` and `let` resulting from ESTree input (#1599)

v5.39.2

- Fix crash when parsing bare `yield` inside a template string.

- Update internally used acorn version requirement

v5.39.1

- Fix bitwise operations that could mix `BigInt` and `number`

v5.39.0

- Remove unnecessary `console.assert` calls (#1590)

v5.38.2

- internal: Flatten inheritance tree for object/class members

v5.38.1

- Fix inlining non-call expressions into an `optional_call?.()`

was submitted, which was nice, but since it wasn't a pure commonJS approach I decided to go with my own solution)

v3.16.0

- No longer leaves names like Array or Object or window as a SimpleStatement (statement which is just a single expression).
- Add support for sections sourcemaps (IndexedSourceMapConsumer)
- Drops node.js v4 and starts using commonJS
- Is now built with rollup

v3.15.0

- Inlined spread syntax (`[...[1, 2, 3], 4, 5] => [1, 2, 3, 4, 5]`) in arrays and objects.
- Fixed typo in compressor warning
- Fixed inline source map input bug
- Fixed parsing of template literals with unnecessary escapes (Like `\\a`)

all unquoted property names, instead of reserving quoted property names automatically.

- Fixed parent functions' parameters being shadowed in some cases
- Allowed Terser to run in a situation where there are custom functions attached to Object.prototype
- And more bug fixes, optimisations and internal changes

v3.17.0

- More DOM properties added to –mangle-properties's DOM property list
- Closed issue where if 2 functions had the same argument name, Terser would not inline them together properly
- Fixed issue with `hasOwnProperty.call`
- You can now list files to minify in a Terser config file
- Started replacing `new Array(<number>)` with an array literal
- Started using ES6 capabilities like `Set` and the `includes` method for strings and arrays

v3.16.1

- Fixed issue where Terser being imported with `import` would cause it not to work due to the `__esModule` property. (PR #254

v5.38.0

- Remove `console` method-of-method calls (eg `console.log.apply()`) when `drop_console` option is used (#1585)
- Remove more object spreads, such as `{ ...void !0 }` (#1142)

v5.37.0

- Reserved object properties from chrome extensions (domprops)
- Fix semicolon insertion between a class property without a semicolon `a` and a computed class property `["prop"]`

v5.36.0

- Support import attributes `with` syntax

v5.35.0

- Ensure parent directory exists when using –output on CLI (#1530)

v5.34.1

- bump the rollup devDependency to disable CVE warnings (Terser was not affected)

v5.34.0

- internal: stop assigning properties to objects they don't belong in
- internal: run compress tests in parallel
- `drop_console`: emit an empty function if the return value of `console.METHOD(...)` may be called.

v5.33.0

- `reduce_vars` improved when dealing with hoisted function definitions (#1544)

- Default values of functions are now correctly converted from Mozilla AST to Terser AST
- JSON ⊂ ECMAScript spec (if you don't know what this is you don't need to)
- Export AST_* classes to library users
- Fixed issue with `collapse_vars` when functions are created with the same name as a variable which already exists
- Added `MutationObserverInit` (Object with options for initialising a mutation observer) properties to the DOM property list
- Custom `Error` subclasses are now internally used instead of old-school Error inheritance hacks.
- Documentation fixes
- Performance optimizations

v4.0.0

- **breaking change**: The `variables` property of all scopes has become a standard JavaScript `Map` as opposed to the old bespoke `Dictionary` object.
- Typescript definitions were fixed
- `terser --help` was fixed
- The public interface was cleaned up
- Fixed optimisation of `Array` and `new Array`
- Added the `keep_quoted=strict` mode to mangle_props, which behaves more like Google Closure Compiler by mangling

v4.1.0

- Internal functions were replaced by `Object.assign`, `Array.prototype.some`, `Array.prototype.find` and `Array.prototype.every`.
- A serious issue where some ESM-native code was broken was fixed.
- Performance improvements were made.
- Support for BigInt was added.
- Inline efficiency was improved. Functions are now being inlined more proactively instead of being inlined only after another Compressor pass.

v4.0.2

(Hotfix release. Reverts unmapped segments PR #342, which will be put back on Terser when the upstream issue is resolved)

v4.0.1

- Collisions between the arguments of inlined functions and names in the outer scope are now being avoided while inlining
- Unmapped segments are now preserved when compressing a file which has source maps

v5.32.0

- `import("module")` can now be input and output from ESTree AST (#1557)
- `BigInt` literals can now be input and output from ESTree AST (#1555)
- `typeof` an object or array (`typeof {}` and `typeof []`) can now be statically evaluated. (#1546)

v5.31.6

- Retain side effects in a `case` when the expression is a sequence (comma) expression

v5.31.5

- Revert v5.31.4, which created mysterious issues #1548, #1549

v5.31.4 (reverted)

- drop_unused: drop classes which only have side effects in the `extends` part

v5.31.3

- drop_unused: drop unused parameters from IIFEs in some more situations.

v5.31.2

- drop_unused: scan variables in self-referential class declarations that contain side effects.
- Don’t add parens to arrow function when it’s the default for an argument (#1540)
- Update domprops (#1538)

v5.31.1

- Allow drop-unused to drop the whole assignment (not just the assigned name) in more situations, in order to avoid duplication of long strings.

v4.1.4

- Fixed a crash when inlining a function into somewhere else when it has interdependent, non-removable variables.

v4.1.3

- Several issues with the `reduce_vars` option were fixed.
- Starting this version, we only have a dist/bundle.min.js

v4.1.2

- The hotfix was hotfixed

v4.1.1

- Fixed a bug where toplevel scopes were being mixed up with lambda scopes

- Functions can no longer be inlined into a place where they're going to be compared with themselves.
- reduce_funcs option is now legacy, as using reduce_vars without reduce_funcs caused some weird corner cases. As a result, it is now implied in reduce_vars and can't be turned off without turning off reduce_vars.
- Bug which would cause a random stack overflow has now been fixed.

v4.2.0

- When the source map URL is `inline`, don't write it to a file.
- Fixed output parens when a lambda literal is the tag on a tagged template string.
- The `mangle.properties.undeclared` option was added. This enables the property mangler to mangle properties of variables which can be found in the name cache, but whose properties are not known to this Terser run.
- The v8 bug where the toString and source representations of regexes like `RegExp("\\\n")` includes an actual newline is now fixed.
- Now we're guaranteed to not have duplicate comments in the output
- Domprops updates

v5.31.0

- Sync up property mangler exceptions with current contents of Firefox and Chrome environments
- Add more webcomponent properties to property mangler exceptions (#1525)
- Drop non-nullish constants in `...spreads` in objects (#1141)

v5.30.4

- Fix parsing `#private in ...` when next to other operators

v5.30.3

- Fix precedence of `#private in ...` operator

v5.30.2

- Avoid optimizations inside computed keys, because they can cause js-engine-specific bugs.

v5.30.1

- Removed useless \ escapes for non-ascii characters
- Make modern identifier characters quoted for older environments (#1512)

v5.30.0

- Improve removal of classes referring to themselves

v5.29.2

- Make sure ‘computed_props’ creates string keys
- Take into account the evaluated size when inlining

v5.29.1

- fix optimisation of all-bits mask check

v4.3.1

- Fixed an issue from 4.3.0 where any block scope within a for loop erroneously had its parent set to the function scopee
- Fixed an issue where compressing IIFEs with argument expansions would result in some parameters becoming undefined
- addEventListener options argument’s properties are now part of the DOM properties list.

v4.3.0

- Do not drop computed object keys with side effects
- Functions passed to other functions in calls are now wrapped in parentheses by default, which speeds up loading most modules
- Objects with computed properties are now less likely to be hoisted
- Speed and memory efficiency optimizations
- Fixed scoping issues with `try` and `switch`

v4.2.1

- Minor refactors
- Fixed a bug similar to #369 in collapse_vars

v4.3.3

- Fixed a problem where parsing template strings would mix up octal notation and a slash followed by a zero representing a null character.
- Started accepting the name `async` in destructuring arguments with default value.
- Now Terser takes into account side effects inside class `extends` clauses.
- Added parens whenever there's a comment between a return statement and the returned value, to prevent issues with ASI.
- Stopped using raw RegExp objects, since the spec is going to continue to evolve. This ensures Terser is able to process new, unknown RegExp flags and features. This is a breaking change in the AST node AST_RegExp.

v4.3.2

- Typescript typing fix
- Ensure that functions can't be inlined, by reduce_vars, into places where they're accessing variables with the same name, but from somewhere else.

v5.29.0

- Re-releases previously reverted 5.28.0
- Fix crash while optimizing some bitwise ops
- (internal) Remove needless wrapper for from_moz (#1499)

v5.28.1

(hotfix release) - Reverts v5.28.0

v5.28.0

- Optimise redundant or shrinkable bitwise operations (`|`, `^`, `&`, `>>`, `<<`)
- Evaluate some `BigInt` math operations

v5.27.2

- Recognise `this` as a reference to the surrounding class in `drop_unused`. Closes #1472

v5.27.1

- Fixed case where `collapse_vars` inlines `await` expressions into non-async functions.

v5.27.0

- Created `minify_sync()` alternative to `minify()` since there's no async code left.

v5.26.0

- Do not take the `/*#__PURE__*/` annotation into account when the `side_effects` compress option is off.
- The `preserve_annotations` option now automatically opts annotation comments in, instead of requiring the `comments` option to be configured for this.
- Refuse to parse empty parenthesized expressions (`()`)

v5.25.0

- Regex properties added to reserved property mangler (#1471)

v4.3.6

(crash hotfix)

v4.3.5

- Fixed an issue with DOS line endings strings separated by `\` and a new line.
- Improved fix for the output size regression related to unused references within the extends section of a class.
- Variable names of anonymous functions (eg: `const x = () => { ... }` or `var func = function () {...}`) are now preserved when keep_fnames is true.
- Fixed performance degradation introduced for large payloads in v4.2.0

v4.3.4

- Fixed a regression where the output size was increased when unused classes were referred to in the extends clause of a class.
- Small typescript typings fixes.
- Comments with `@preserve`, `@license`, `@cc_on` as well as comments starting with `/*!` and `/**!` are now preserved by default.

- Fixed invalid output caused by the creation of empty sequences internally
- Scopes are now updated when scopes are inlined into them

v4.3.9

- Fixed issue with mangle's `keep_fnames` option, introduced when adding code to keep variable names of anonymous functions

v4.3.8

- Typescript typings fix

v4.3.7

- Parsing of regex options in the CLI (which broke in v4.3.5) was fixed.
- typescript definition updates

- `pure_new` option added to drop unused `new` expressions.

v5.24.0

- Improve formatting performance in V8 by keeping a small work string and a large output string

v5.23.0

- When top_retain will keep a variable assignment around, inline the assignee when it's shorter than the name (#1434)
- Remove empty class `static {}` blocks.

v5.22.0

- Do not `unsafe`ly shorten expressions like a?.toString() when they're conditional.
- Avoid running drop_unused in nodes that aren't scopes. Fixes a rare crash.
- When 'module' is enabled, assume strict mode when figuring out scopes.

v5.21.0

- Do not inline functions that would be retained in the toplevel (as this would cause code duplication).
- Fix precedence of arrow function and ternary operator when formatting output.

v5.20.0

- Passing `minify()` zero files will now throw a clean exception (#1450)
- `drop_console` supports passing in an array of `console.*` method names (#1445)
- New DOM properties from the WebGPU API have been added for use in the property mangler (#1436)
- Internal code simplification (#1437)

v5.19.4

- Prevent creating very deeply nested ternaries from a long list of `if..return`
- Prevent inlining classes into other functions, to avoid constructors being compared.

v4.4.0

- Added `/*#__INLINE__*/` and `/*#__NOINLINE__*/` annotations for calls. If a call has one of these, it either forces or forbids inlining.

v4.3.11

- Fixed a problem where `window` was considered safe to access, even though there are situations where it isn't (Node.js, workers…)
- Fixed an error where `++` and `--` were considered side-effect free
- `Number(x)` now needs both `unsafe` and and `unsafe_math` to be compressed into `+x` because `x` might be a `BigInt`
- `keep_fnames` now correctly supports regexes when the function is in a variable declaration

v4.3.10

- Fixed syntax error when repeated semicolons were encountered in classes

v4.4.3

- Number and BigInt parsing has been fixed
- `/*#__INLINE__*/` annotation fixed for arrow functions with non-block bodies.
- Functional tests have been added, using this repository.
- A memory leak, where the entire AST lives on after compression, has been plugged.

v4.4.2

- Fixed a problem with inlining identity functions

v4.4.1

note: This introduced a feature, therefore it should have been a minor release.

- Fixed a crash when `unsafe` was enabled.
- An issue has been fixed where `let` statements might be collapsed out of their scope.
- Some error messages have been improved by adding quotes around variable names.

v5.19.3

- Fix side effect detection of `optional?.chains`.
- Add roundRect to domprops.js (#1426)

v5.19.2

- fix performance hit from avoiding HTML comments in the output

v5.19.1

- Better avoid outputting `</script>` and HTML comments.
- Fix unused variables in class static blocks not being dropped correctly.
- Fix sourcemap names of methods that are `async` or `static`

v5.19.0

- Allow `/*@__MANGLE_PROP__*/` annotation in `object.property`, in addition to property declarations.

v5.18.2

- Stop using recursion in hoisted defuns fix.

v5.18.1

- Fix major performance issue caused by hoisted defuns' scopes bugfix.

v5.18.0

- Add new `/*@__MANGLE_PROP__*/` annotation, to mark properties that should be mangled.

v5.17.7

- Update some dependencies
- Add consistent sorting for `v` RegExp flag
- Add `inert` DOM attribute to domprops

v4.6.0

- Fixed issues with recursive class references.
- BigInt evaluation has been prevented, stopping Terser from evaluating BigInts like it would do regular numbers.
- Class property support has been added

v4.5.1

(hotfix release)

- Fixed issue where `() => ({})[something]` was not parenthesised correctly.

v4.5.0

- Inlining has been improved
- An issue where keep_fnames combined with functions declared through variables was causing name shadowing has been fixed
- You can now set the ES version through their year
- The output option `keep_numbers` has been added, which prevents Terser from turning `1000` into `1e3` and such
- Internal small optimisations and refactors

- Typescript typings improvements
- Optimizations while looking for surrogate pairs in strings

v4.6.3

- Annotations such as `/*#__NOINLINE__*/` and `/*#__PURE__*/` may now be preserved using the `preserve_annotations` output option
- A TypeScript definition update for the `keep_quoted` output option.

v4.6.2

- A bug where functions were inlined into other functions with scope conflicts has been fixed.
- `/*#__NOINLINE__*/` annotation fixed for more use cases where inlining happens.

v4.6.1

- Fixed an issue where a class is duplicated by reduce_vars when there's a recursive reference to the class.

v5.17.6

- Fixes to mozilla AST input and output, for class properties, private properties and static blocks
- Fix outputting a shorthand property in quotes when safari10 and ecma=2015 options are enabled
- `configurable` and `enumerable`, used in Object.defineProperty, added to domprops (#1393)

v5.17.5

- Take into account the non-deferred bits of a class, such as static properties, while dropping unused code.

v5.17.4

- Fix crash when trying to negate a class (`!class{}`)
- Avoid outputting comments between `yield`/`await` and its argument
- Fix detection of left-hand-side of assignment, to avoid optimizing it like any other expression in some edge cases

v5.17.3

- Fix issue with trimming a static class property's contents accessing the class as `this`.

v5.17.2

- Be less conservative when detecting use-before-definition of `var` in hoisted functions.
- Support unusual (but perfectly valid) initializers of for-in and for-of loops.
- Fix issue where hoisted function would be dropped if it was after a `continue` statement

v5.17.1

- Fix evaluating `.length` when the source array might've been mutated

v4.6.6

(hotfix release)

- Reverted code to 4.6.4 to allow for more time to investigate an issue.

v4.6.5 (REVERTED)

- Improved compress performance through using a new method to see if two nodes are equivalent, instead of printing them to a string.

v4.6.4

- The `"some"` value in the `comments` output option now preserves `@lic` and other important comments when using `//`
- `</script>` is now better escaped in regex, and in comments, when using the `inline_script` output option
- Fixed an issue when transforming `new RegExp` into `/.../` when slashes are included in the source
- `AST_Node.prototype.constructor` now exists, allowing for easier debugging of crashes
- Multiple if statements with the same consequents are now collapsed

v4.6.9

- Check if block scopes actually exist in blocks

v4.6.8

- Take into account “executed bits” of classes like static properties or computed keys, when checking if a class evaluation might throw or have side effects.

v4.6.7

- Some new performance gains through a `AST_Node.size()` method which measures a node’s source code length without printing it to a string first.
- An issue with setting `--comments` to `false` in the CLI has been fixed.
- Fixed some issues with inlining
- `unsafe_symbols` compress option was added, which turns `Symbol("name")` into just `Symbol()`
- Brought back compress performance improvement through the `AST_Node.equivalent_to(other)` method (which was reverted in v4.6.6).

v5.17.0

- Drop vestigial `= undefined` default argument in IIFE calls (#1366)
- Evaluate known arrays’ `.length` property when statically determinable
- Add `@__KEY__` annotation to mangle string literals (#1365)

v5.16.9

- Fix parentheses in output of optional chains (`a?.b`) (#1374)
- More documentation on source maps (#1368)
- New `lhs_constants` option, allowing to stop Terser from swapping comparison operands (#1361)

v5.16.8

- Become even less conservative around function definitions for `reduce_vars`
- Fix parsing context of `import.meta` expressions such that method calls are allowed

v5.16.6

- Become less conservative with analyzing function definitions for `reduce_vars`
- Parse `import.meta` as a real AST node and not an `object.property`

v5.16.5

- Correctly handle AST transform functions that mutate children arrays
- Don't mutate the options object passed to Terser (#1342)
- Do not treat BigInt like a number

v5.16.4

- Keep `(defaultArg = undefined) => ...`, because default args don't count for function length
- Prevent inlining variables into `?.` optional chains
- Avoid removing unused arguments while transforming
- Optimize iterating AST node lists
- Make sure `catch` and `finally` aren't children of `try` in the AST

- Printing of unicode identifiers has been improved

v4.6.11

- Read unused classes' properties and method keys, to figure out if they use other variables.
- Prevent inlining into block scopes when there are name collisions
- Functions are no longer inlined into parameter defaults, because they live in their own special scope.
- When inlining identity functions, take into account the fact they may be used to drop `this` in function calls.
- Nullish coalescing operator (`x ?? y`), plus basic optimization for it.
- Template literals in binary expressions such as `+` have been further optimized

v4.6.10

- Do not use reduce_vars when classes are present

- Fixed bug where `yield` wasn't considered a valid property key in generators.

v4.7.0

- A bug was fixed where an arrow function would have the wrong size
- `arguments` object is now considered safe to retrieve properties from (useful for `length`, or `0`) even when `pure_getters` is not set.
- Fixed erroneous `const` declarations without value (which is invalid) in some corner cases when using `collapse_vars`.

v4.6.13

- Fixed issue where ES5 object properties were being turned into ES6 object properties due to more lax unicode rules.
- Fixed parsing of BigInt with lowercase `e` in them.

v4.6.12

- Fixed subtree comparison code, making it see that `[1,[2, 3]]` is different from `[1, 2, [3]]`

- Use modern unicode property escapes (`\p{...}`) to parse identifiers when available

v5.16.3

- Ensure function definitions, don't assume the values of variables defined after them.

v5.16.2

- Fix sourcemaps with non-ascii characters (#1318)
- Support string module name and export * as (#1336)
- Do not move `let` out of `for` initializers, as it can change scoping
- Fix a corner case that would generate the invalid syntax `if (something) let x` ("let" in braceless if body)
- Knowledge of more native object properties (#1330)
- Got rid of Travis (#1323)
- Added semi-secret `asObject` sourcemap option to typescript defs (#1321)

v5.16.1

- Properly handle references in destructurings (`const { [reference]: val } = ...`)
- Allow parsing of `.#privatefield` in nested classes
- Do not evaluate operations that return large strings if that would make the output code larger
- Make `collapse_vars` handle block scope correctly
- Internal improvements: Typos (#1311), more tests, small-scale refactoring

v5.16.0

- Disallow private fields in object bodies (#1011)
- Parse `#privatefield in object` (#1279)
- Compress `#privatefield in object`

v5.15.1

- Fixed missing parentheses around optional chains
- Avoid bare `let` or `const` as the bodies of `if` statements (#1253)
- Small internal fixes (#1271)

v5.0.0-beta.0

- BREAKING: `minify()` is now async and rejects a promise instead of returning an error.
- BREAKING: Internal AST is no longer exposed, so that it can be improved without releasing breaking changes.
- BREAKING: Lowest supported node version is 10
- BREAKING: There are no more warnings being emitted
- Module is now distributed as a dual package - You can `import` and `require()` too.
- Inline improvements were made

---

v4.8.1 (backport)

- Security fix for RegExps that should not be evaluated (regexp DDOS)

v4.8.0

- Support for numeric separators (`million = 1_000_000`) was added.
- Assigning properties to a class is now assumed to be pure.

v5.2.0

- Optional chaining syntax is now supported.
- Consecutive await expressions don't have unnecessary parens
- Taking the variable name's length (after mangling) into consideration when deciding to inline

v5.1.0

- `import.meta` is now supported
- Typescript typings have been improved

v5.0.0

- `in` operator now taken into account during property mangle.
- Fixed infinite loop in face of a reference loop in some situations.
- Kept exports and imports around even if there's something which will throw before them.
- The main exported bundle for commonjs, dist/bundle.min.js is no longer minified.

- Avoid inlining a class twice and creating two equivalent but `!==` classes.

v5.15.0

- Basic support for ES2022 class static initializer blocks.
- Add `AudioWorkletNode` constructor options to domprops list (#1230)
- Make identity function inliner not inline `id(...expandedArgs)`

v5.14.2

- Security fix for RegExps that should not be evaluated (regexp DDOS)
- Source maps improvements (#1211)
- Performance improvements in long property access evaluation (#1213)

v5.14.1

- keep_numbers option added to TypeScript defs (#1208)
- Fixed parsing of nested template strings (#1204)

v5.14.0

- Switched to @jridgewell/source-map for sourcemap generation (#1190, #1181)
- Fixed source maps with non-terminated segments (#1106)
- Enabled typescript types to be imported from the package (#1194)
- Extra DOM props have been added (#1191)
- Delete the AST while generating code, as a means to save RAM

v5.13.1

- Removed self-assignments (`varname=varname`) (closes #1081)
- Separated inlining code (for inlining things into references, or removing IIFEs)
- Allow multiple identifiers with the same name in `var` destructuring (eg `var { a, a } = x`) (#1176)

v5.13.0

- All calls to eval() were removed (#1171, #1184)
- `source-map` was updated to 0.8.0-beta.0 (#1164)

v5.3.1

- An issue with destructuring declarations when `pure_getters` is enabled has been fixed
- Fixed a crash when chain expressions need to be shallowly compared
- Made inlining functions more conservative to make sure a function that contains a reference to itself isn't moved into a place that can create multiple instances of itself.

v5.3.0

- Fixed a crash when compressing object spreads in some cases
- Fixed compiletime evaluation of optional chains (caused typeof a?.b to always return "object")
- domprops has been updated to contain every single possible prop

v5.2.1

- The parse step now doesn't accept an `ecma` option, so that all ES code is accepted.
- Optional dotted chains now accept keywords, just like dotted expressions (`foo?.default`)

v5.3.4

- Fixed a crash when hoisting (with `hoist_vars`) a destructuring variable declaration

v5.3.3

- `source-map` library has been updated, bringing memory usage and CPU time improvements when reading input source maps (the SourceMapConsumer is now WASM based).
- The `wrap_func_args` option now also wraps arrow functions, as opposed to only function expressions.

v5.3.2

- Prevented spread operations from being expanded when the expanded array/object contains getters, setters, or array holes.
- Fixed *very* slow self-recursion in some cases of removing extraneous parentheses from + operations.

- NavigatorUAData was added to domprops to avoid property mangling (#1166)

v5.12.1

- Fixed an issue with function definitions inside blocks (#1155)
- Fixed parens of `new` in some situations (closes #1159)

v5.12.0

- `TERSER_DEBUG_DIR` environment variable
- @copyright comments are now preserved with the comments="some" option (#1153)

v5.11.0

- Unicode code point escapes (`\u{abcde}`) are not emitted inside RegExp literals anymore (#1147)
- acorn is now a regular dependency

v5.10.0

- Massive optimization to max_line_len (#1109)
- Basic support for import assertions
- Marked ES2022 Object.hasOwn as a pure function
- Fix `delete optional?.property`
- New CI/CD pipeline with github actions (#1057)
- Fix reordering of switch branches (#1092), (#1084)
- Fix error when creating a class property called `get`
- Acorn dependency is now an optional peerDependency
- Fix mangling collision with exported variables (#1072)
- Fix an issue with `return someVariable = (async () => { ... })()` (#1073)

v5.9.0

- Collapsing switch cases with the same bodies (even if they're not next to each other) (#1070).
- Fix evaluation of optional chain expressions (#1062)
- Fix mangling collision in ESM exports (#1063)
- Fix issue with mutating function objects after a second pass (#1047)
- Fix for inlining object spread `{ ...obj }` (#1071)
- Typescript typings fix (#1069)

v5.3.8

- Restore node 13 support

v5.3.7

Hotfix release, fixes package.json "engines" syntax

v5.3.6

- Fixed parentheses when outputting `??` mixed with `||` and &&
- Improved hygiene of the symbol generator

v5.3.5

- Avoid moving named functions into default exports.
- Enabled transform() for chain expressions. This allows AST transformers to reach inside chain expressions.

- Private properties, getters and setters have been added in #913 and some more commits
- Docs improvements: #896, #903, #916

v5.5.1

- Fixed object properties with unicode surrogates on safari.

v5.5.0

- Fixed crash when inlining uninitialized variable into template string.
- The sourcemap for dist was removed for being too large.

v5.4.0

- Logical assignment
- Change `let x = undefined` to just `let x`
- Removed some optimizations for template strings, placing them behind `unsafe` options. Reason: adding strings is not equivalent to template strings, due to valueOf differences.
- The AST_Token class was slimmed down in order to use less memory.

v5.8.0

- Fixed shadowing variables while moving code in some cases (#1065)
- Stop mangling computed & quoted properties when keep_quoted is enabled.
- Fix for mangling private getter/setter and .#private access (#1060, #1068)
- Array.from has a new optimization when the unsafe option is set (#737)
- Mangle/propmangle let you generate your own identifiers through the nth_identifier option (#1061)
- More optimizations to switch statements (#1044)

v5.7.2

- Fixed issues with compressing functions defined in `global_defs` option (#1036)
- New recipe for using Terser in gulp was added to RECIPES.md (#1035)
- Fixed issues with `??` and `?.` (#1045)
- Future reserved words such as `package` no longer require you to disable strict mode to be used as names.
- Refactored huge compressor file into multiple more focused files.

- Avoided unparenthesized `in` operator in some for loops (it breaks parsing because of for..in loops)
- Improved documentation (#1021, #1025)
- More type definitions (#1021)

v5.7.1

- Avoided collapsing assignments together if it would place a chain assignment on the left hand side, which is invalid syntax (`a?.b = c`)
- Removed undefined from object expansions (`{ ...void 0 }` -> `{}`)
- Fix crash when checking if something is nullish or undefined (#1009)
- Fixed comparison of private class properties (#1015)
- Minor performance improvements (#993)
- Fixed scope of function defs in strict mode (they are block scoped)

v5.7.0

- Several compile-time evaluation and inlining fixes
- Allow `reduce_funcs` to be disabled again.
- Add `spidermonkey` options to parse and format (#974)

- Accept `{get = "default val"}` and `{set = "default val"}` in destructuring arguments.
- Change package.json export map to help require.resolve (#971)
- Improve docs
- Fix `export default` of an anonymous class with `extends`

v5.6.1

- Mark assignments to the `.prototype` of a class as pure
- Parenthesize `await` on the left of `**` (while accepting legacy non-parenthesised input)
- Avoided outputting NUL bytes in optimized RegExps, to stop the output from breaking other tools
- Added `exports` to domprops (#939)
- Fixed a crash when spreading `...this`
- Fixed the computed size of arrow functions, which improves their inlining

v5.6.0

- Added top-level await
- Beautify option has been removed in #895