Add comment delims to grammar declaration

Status

Proposed

Summary

Grammars currently only sort of declare their comment delims. E.g., TextMate JavaScript will say `'commentStart': '// '` (and not even in the same file as the grammar), and Tree-sitter says `comments: start: '// '` (in the same file; better). However, it is impossible to tell that `/* */` delimits JS block comments. This RFC is to request a grammar property to declare all comment delims a language supports.

Motivation

It can be useful for a package to be able to treat comment characters abstractly.

My specific need is for the resolution of `BLOCK_COMMENT_START` and similar variables for LSP snippets. There is currently no way I know of to resolve this without hardcoding it for each language.

Explanation

This RFC is to have both line and block delims a property of the grammar. E.g.,

```
# javascript.cson
comments:
    line: '//'
    start: '/*'
    end: '*/'
```

The lack of a property would indicate the grammar does not have it (e.g., HTML would omit the `line` property), or that a grammar has not been updated to comply with this spec. If the latter, it may be possible to partially extract this information from the existing implementations (e.g., even now we can tell that JS has `//` for line comment and HTML has `<!-- -->` for block comments).

This is similar to the current Tree-sitter grammars, but they currently mix line and block in the `start` key depending on if the language has line comments (so HTML has `start: '<!--', end: '-->'`, but JS has `start: '//'`).

Drawbacks

Many community grammars would not get this property added to them. However, this feature can be considered a strict enhancement of the current status, and non compliant grammars can be accounted for. E.g., if a grammar still declares `start: '//'` but doesn't have an `end` property, then it can be reinterpreted

as a line delim. Additionally, users would be quick to raise an issue here (:frowning_face:) or on the language repo (:slightly_smiling_face:) if they are trying to use a feature that relies on this and it doesn't work.

Rationale and alternatives

Why is this approach the best in the space of possible approaches? Tying all language specific data to the language file makes intuitive sense. This is stuff that will not change based on what the user wants (and already is tied directly to Tree-sitter language files).

What other approaches have been considered and what is the rationale for not choosing them? It's possible to use the settings approach like for TextMate grammars. I find this unnecessarily separated though, especially for something like comment delims which shouldn't rely on what the user fancies.

However, I'm not set on requiring the TextMate grammars to have it in the file (doing so would require an update on the First mate side too*). It can still work in the settings file. This would also support the possible language that has multiple delim characters (if it exists), letting the user set their choice.

* Maybe First mate should just add all properties from the file to the grammar it constructs, instead of a whitelist? It would save headaches around enhancing future grammar features.

What is the impact of not doing this? Getting the snippet variables working would require hard coding them for each language, which is impossible to do completely.

Unresolved questions

What unresolved questions do you expect to resolve through the RFC process before this gets merged?

What unresolved questions do you expect to resolve through the implementation of this feature before it is released in a new version of Atom?

What related issues do you consider out of scope for this RFC that could be addressed in the future independently of the solution that comes out of this RFC? What I would like is then for public TextEditor methods `getCommentDelims` and `getCommentDelimsForPoint`, which returns all the correct delims for the root grammar, or the one at the given point (accounting for embedded grammars ... though could be weird when the embedded grammar is only something like TODO or SQL syntax).

However, this future enhancement is not necessary for the current RFC. This RFC is about getting comment delim information tied to the Grammar object and is independant of any attempt to handle this information.