Migrating 0.x to 1.x

Parser

0.x can be mostly translated into 1.x one way or another. The idea behind the new config structure is to handle only the most common cases, and provide the fallback for alternative implementation.

`dotted_names: boolean`

> By default dotted names like `name.subname.subsubname` will be expanded into nested sections, this can be prevented by passing opts.dotted_names = false.

Removed This feature is removed but still can be done on top of the `parse()` output. Please post a request or contribute a PR if you need it.

`trim: boolean`

> Set this to false to avoid the default of trimming whitespace at the start and end of each line.

In the new parser all original spacing is kept along with comment lines in `.source`. Description lines are joined together depending on `spacing` option

New option:

- `spacing: "compact"` lines concatenated with a single space and no line breaks
- `spacing: "preserve"` keeps line breaks and space around as is. Indentation space counts from `*` delimiter or from the start of the line if the delimiter is omitted
- `spacing: (lines: Line[]) => string` completely freeform joining strategy, since all original spacing can be accessed, there is no limit to how this can be implemented. See primitives.ts and spacer.ts

`join: string | number | boolean`

> If the following lines of a multiline comment do not start with a star, `join` will have the following effect on tag source (and description) when joining the lines together:
>
> - If a string, use that string in place of the leading whitespace (and avoid newlines).
> - If a non-zero number (e.g., 1), do no trimming and avoid newlines.
> - If undefined, false, or 0, use the default behavior of not trimming but adding a newline.
> - Otherwise (e.g., if join is true), replace any leading whitespace with a single space and avoid newlines.

Note that if a multi-line comment has lines that start with a star, these will be appended with initial whitespace as is and with newlines regardless of the join setting.

See the `spacing` option above, all the variations can be fine-tunned with `spacing: (lines: Line[]) => string`

`fence: string | RegExp | ((source: string) => boolean)`

Set to a string or regular expression to toggle state upon finding an odd number of matches within a line. Defaults to “‘.

If set to a function, it should return true to toggle fenced state; upon returning true the first time, this will prevent subsequent lines from being interpreted as starting a new jsdoc tag until such time as the function returns true again to indicate that the state has toggled back.

This is mostly kept the same

New optoins:

- `fence: '```'` same as 0.x
- `fencer: (source: string) => boolean` same as 0.x, see parser/block-parser.ts

`parsers: Parser[]`

In case you need to parse tags in different way you can pass opts.parsers = [parser1, . . . , parserN], where each parser is function name(str:String, data:Object):{source:String, data:Object}. . . .

New options:

- `tokenizers: []Tokenizer` is a list of functions extracting the `tag`, `type`, `name` and `description` tokens from this string. See parser/spec-parser.ts and primitives.ts

Default tokenizers chain is

```
[
  tagTokenizer(),
  typeTokenizer(),
  nameTokenizer(),
  descriptionTokenizer(getSpacer(spacing)),
]
```

where

```
type Tokenizer = (spec: Spec) => Spec

interface Spec {
```

```
  tag: string;
  name: string;
  default?: string;
  type: string;
  optional: boolean;
  description: string;
  problems: Problem[];
  source: Line[];
}
```

chain starts with blank `Spec` and each tokenizer fulfills a piece using `.source` input

Stringifier

> One may also convert comment-parser JSON structures back into strings using the stringify method (stringify(o: (object|Array) [, opts: object]): string). ...

Stringifier config follows the same strategy – a couple of common cases, and freeform formatter as a fallback

New Options:

- `format: "none"` re-assembles the source with original spacing and delimiters preserved
- `format: "align"` aligns tag, name, type, and descriptions into fixed-width columns
- `format: (tokens: Tokens) => string[]` do what you like, resulting lines will be concatenated into the output. Despite the simple interface, this can be turned into a complex stateful formatter, see `"align"` implementation in transforms/align.ts

Stream

Work in progress