How does TypeScript formatting work?

To format code you need to have a formatting context and a `SourceFile`. The formatting context contains all user settings like tab size, newline character, etc.

The end result of formatting is represented by TextChange objects which hold the new string content, and the text to replace it with.

```
export interface TextChange {
    span: TextSpan; // start, length
    newText: string;
}
```

Internals

Most of the exposed APIs internally are `format*` and they all set up and configure `formatSpan` which could be considered the root call for formatting. Span in this case refers to the range of the sourcefile which should be formatted.

The formatSpan then uses a scanner (either with or without JSX support) which starts at the highest node the covers the span of text and recurses down through the node's children.

As it recurses, `processNode` is called on the children setting the indentation is decided and passed through into each of that node's children.

The meat of formatting decisions is made via `processPair`, the pair here being the current node and the previous node. `processPair` which mutates the formatting context to represent the current place in the scanner and requests a set of rules which can be applied to the items via `createRulesMap`.

There are a lot of rules, which you can find in rules.ts each one has a left and right reference to nodes or token ranges and note of what action should be applied by the formatter.

Where is this used?

The formatter is used mainly from any language service operation that inserts or modifies code. The formatter is not exported publicly, and so all usage can only come through the language server.