PHPStanのエラーをprettyにしようとしている

Natsuki

PHPStan使ってますか

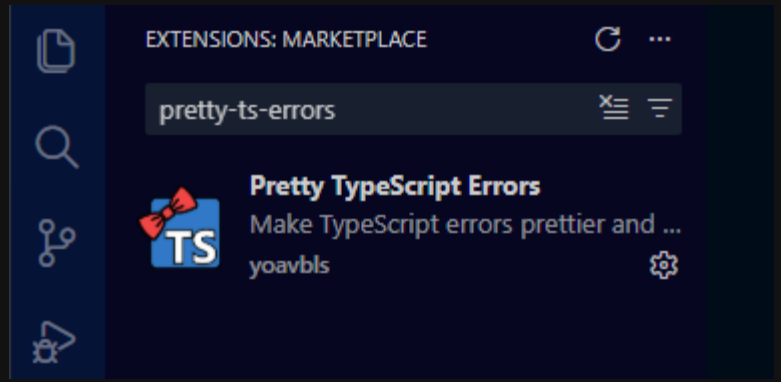
```php
<?php

class User {
  public function __construct(
    public string $name,
    public ?int $age
  ) {}

    public function nextAge(): int {
        return $this->age + 1;
    }
}
```

VSCodeのエラーは見づら過ぎる!!

```php
/**
 * @return array{
 *     id: int,
 *     name: string,
 *     // ...
 * }
 */
function getProfile(): array
{
  return [
    'id' => 123,
    'name' => 'John Doe',
    // ...
  ];
}
```

```
  age: int,
Function buildUserProfile() should return array{id: int, name: string, email: string, age: int, address: array{zip: string,
prefecture: string, city: string, street: string}, tags: array<string>, created_at: string, updated_at: string} but returns
array{id: 1, name: 'Alice', email: 'alice@example.com', age: 20, address: array{zip: '123-4567', prefecture: 'Tokyo', city:
'Chiyoda'}, tags: array{'admin', 'editor'}, created_at: '2022-01-01 00:00:00'}.
💡 Offset 'address' (array{zip: string, prefecture: string, city: string, street: string}) does not accept type array{zip: '123-
4567', prefecture: 'Tokyo', city: 'Chiyoda'}: Array does not have offset 'street'. PHPStan(return.type)
```

pretty-ts-errorsみたいなのが欲しい



エラー文をパスーすればいいのでは(1)

php-yacc
第179回 PHP勉強会@東京でのきんじょうさんの話

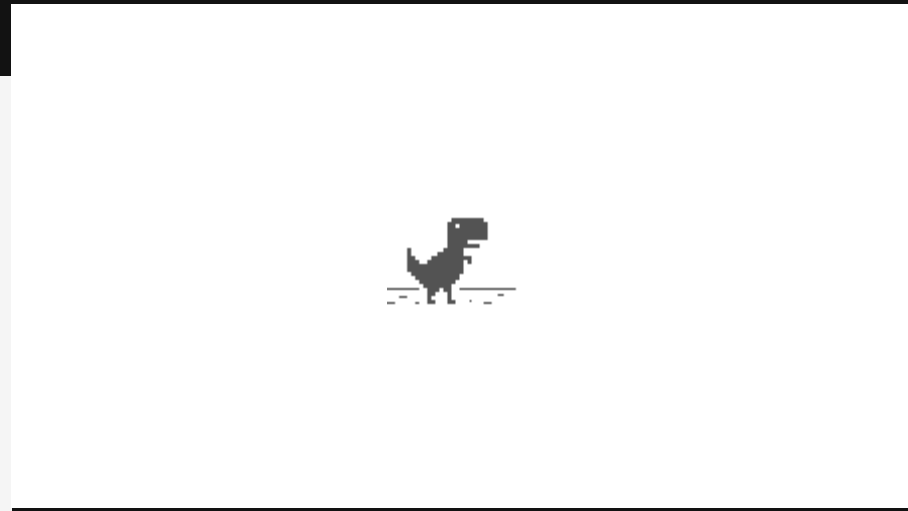
オフラインです

エラー文をパスーすればいいのでは(2)

racc

Umeda.rbにて わいださん のraccの話

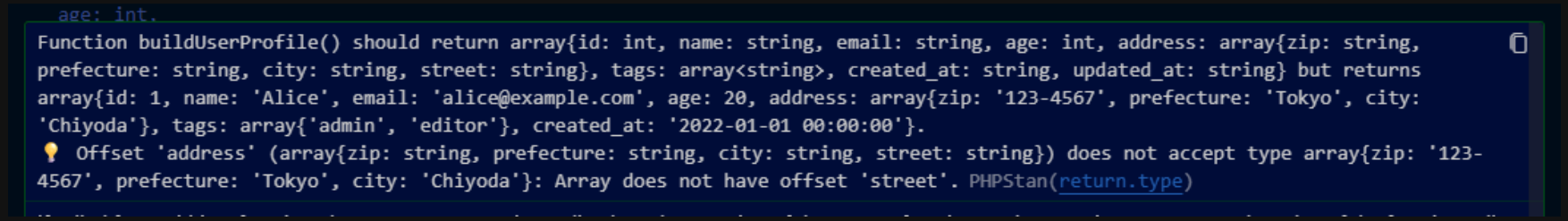
ぺちこん関西2025でのトーク



LRパーサーとか難しいのでなんとなくの理解

パーサーを使わないアプローチ

パーサーは作らずエラーパターンからマッチさせる




```
/Function 関数名 should return 型(array{...}) but returns 型(array{...})./
```

エラーの種類だけ、パターンマッチで拾う

エラーパターンの変更を追従するのが大変 & プラグインの数だけ対応が必要
=> 汎用的にエラー文をパースできた方がいい

ちなみにpretty-ts-errorsは正規表現

```
message
.replaceAll(
  /(is missing the following properties from type\s?)'(.*)': ((?:#?\w+, )*(?:(?!and)\w+)?)/g,
  (_, pre, type, post) =>
    `${pre}${formatTypeBlock("", type, codeBlock)}: <ul>${post
      .split(", ")
      .filter(Boolean)
      .map((prop: string) => `<li>${prop}</li>`)
      .join("")}</ul>`
  )
```

https://github.com/yoavbls/pretty-ts-errors/blob/8527285178e9a88c80ad63fa7f0129192b67bad2/packages/formatter

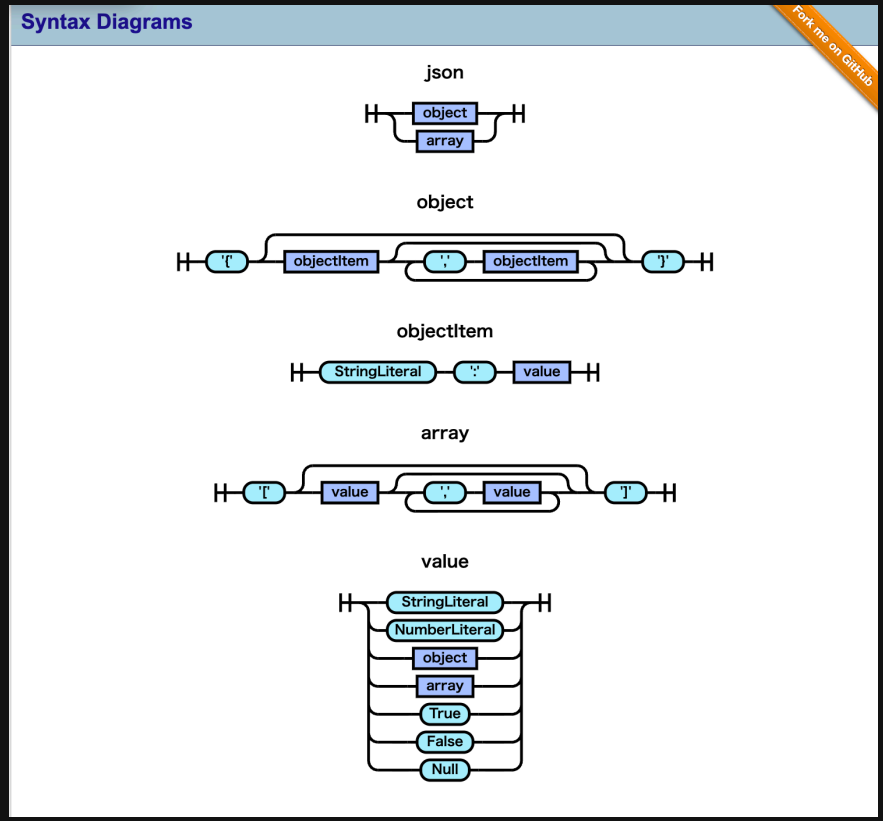
TSはエラーパターンが決まってるからできる

Chevrotain

TypeScript製のパーサージェネレーター

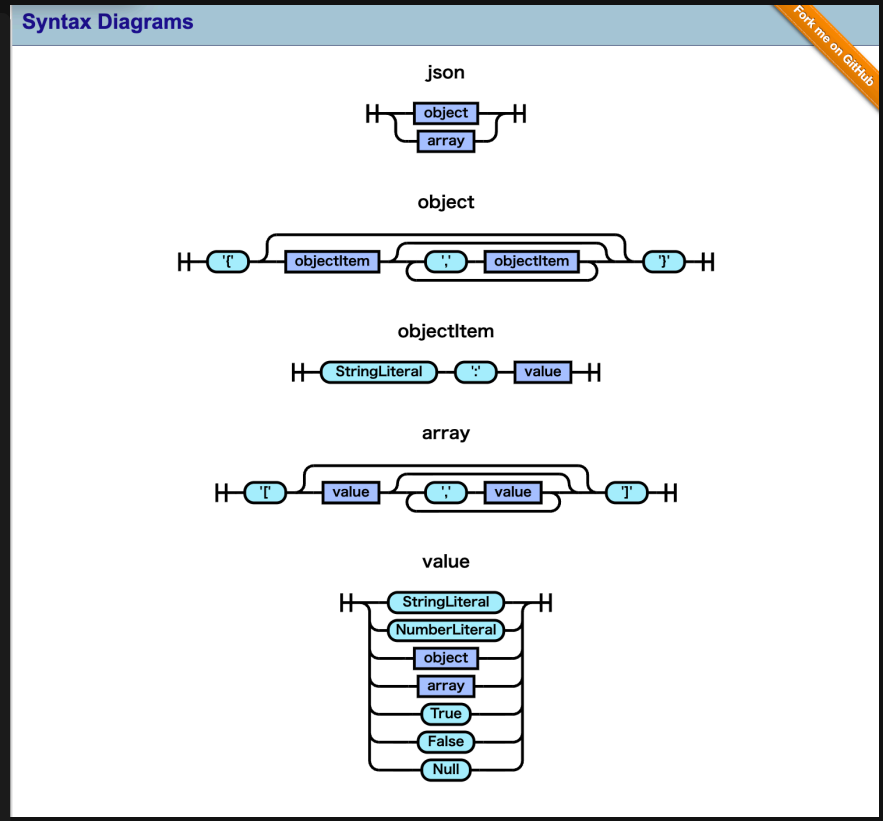
マメジカ

例


Syntax Diagrams
Fork me on GitHub
json
object
array
object
'{'
objectItem
','
objectItem
'}'
objectItem
StringLiteral
':'
value
array
'['
value
','
value
']'
value
StringLiteral
NumberLiteral
object
array
True
False
Null

例

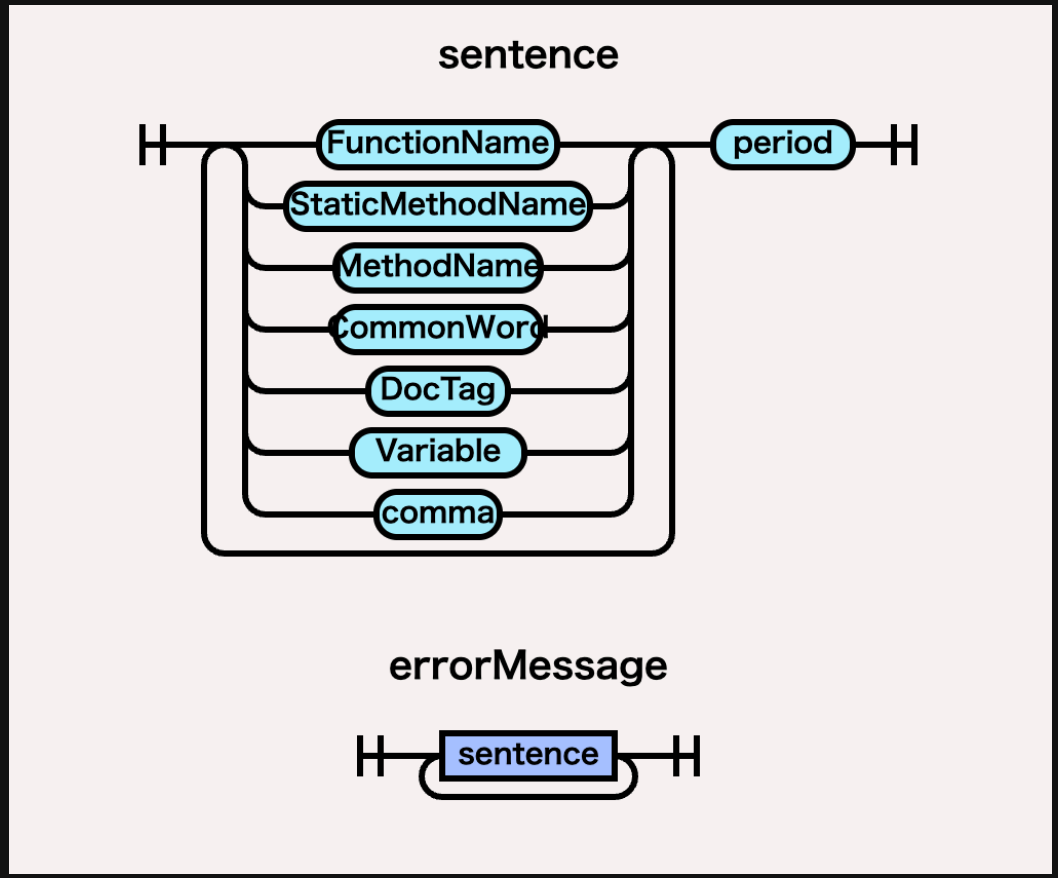
```
class JsonParser extends CstParser {
  constructor() {
    this.RULE("json", () => {
      this.OR([
        { ALT: () => this.SUBRULE(this.object) },
        { ALT: () => this.SUBRULE(this.array) },
      ]);
    });

    this.RULE("object", () => {
      this.CONSUME(LCurly);
      this.MANY_SEP({
        SEP: Comma,
        DEF: () => {
          this.SUBRULE(this.objectItem);
        },
      });
      this.CONSUME(RCurly);
    });

    this.RULE("objectItem", () => {
      this.CONSUME(StringLiteral);
      this.CONSUME(Colon);
      this.SUBRULE(this.value);
```



パーサーの定義の前に字句解析(Lexer)

```
const True = createToken({ name: "True", pattern: /true/ });
const False = createToken({ name: "False", pattern: /false/ });
const Null = createToken({ name: "Null", pattern: /null/ });
const LCurly = createToken({ name: "LCurly", pattern: /{/ });
const RCurly = createToken({ name: "RCurly", pattern: /}/ });
```

phpstan-error-parserを作っています



使い方

```
import { parse } from 'phpstan-error-parser';

const result = parse('PHPDoc tag @mixin contains unresolvable type.')
```

```
[
  {
    type: 'common_word',
    value: 'PHPDoc',
    location: {
      startColumn: 0,
      endColumn: 6,
    },
  },
  {
    type: 'common_word',
    value: 'tag',
    location: {
      startColumn: 7,
      endColumn: 10,
    },
  },
  {
    type: 'doc_tag',
    value: '@mixin',
```

VSCode拡張の組み込み

アプローチは二つ

pretty-ts-errorsのように別拡張として提供

phpstan-vscode自体に組み込む

pretty-ts-errorsとphpstan-vscodeがエラーを表示している仕組み

pretty-ts-errors ⇒ `Diagnostic`

phpstan-vscode ⇒ `HoverProvider`

Diagnostic(診断)

場所とメッセージを渡すとエラー表示してくれる

```
interface Diagnostic {
    range: Range;

    severity?: DiagnosticSeverity;

    source?: string;

    message: string;
}
```

https://microsoft.github.io/language-server-protocol/specifications/lsp/3.17/specification/#diagnostic

phpstan-vscodeはPHPStanの結果をこれでVSCodeに渡している
messageにMarkdownを渡せばリッチな表示ができる？ ⇒ できない🙅

HoverProvider

ホバーイベントに応じてメッセージを返す

```
interface Hover {
    /**
     * The hover's content
     */
    contents: MarkedString | MarkedString[] | MarkupContent;

    /**
     * An optional range is a range inside a text document
     * that is used to visualize a hover, e.g. by changing the background color.
     */
    range?: Range;
}
```

https://microsoft.github.io/language-server-protocol/specifications/lsp/3.17/specification/#textDocument_hover

`Hover`ではcontentsにMarkdownを渡せばリッチな表示ができる❗❗

pretty-ts-errorsでもそれによるDiscussionがある

https://github.com/yoavbls/pretty-ts-errors/discussions/43

2. Allow diagnostics messages to have markdown

> Solves #15, #19, #3 and make us closer to solve: #11, #21, #26, #25

If VSCode allows it we could be able to move the formatting to a language server plugin which will allow us to:

1. Control the order of the original error and the pretty one
2. Hide the original error without losing it in the problems pane in VSCode or for other extensions like Error Lens
3. Support more platforms more easily like Neovim, Visual Studio and maybe even JetBrains IDEs

⬆️ **Upvote or comment here**

#

LSP 3.18 ならDiagnosticにもMarkdownが使える

```
/**
 * The diagnostic's message.
 *
 * @since 3.18.0 - support for MarkupContent. This is guarded by the client
 * capability `textDocument.diagnostic.markupMessageSupport`.
 */
message: string | MarkupContent;
```

https://microsoft.github.io/language-server-protocol/specifications/lsp/3.18/specification/#diagnostic

いつになるか分からないので

別拡張としてphpstan-error-parserでパースして、HoverProviderで実装する予定

おしまい