NAME

TAP::Parser::Result - Base class for TAP::Parser output objects

VERSION

Version 3.26

SYNOPSIS

```
  # abstract class - not meant to be used directly
  # see TAP::Parser::ResultFactory for preferred usage

  # directly:
  use TAP::Parser::Result;
  my $token  = {...};
  my $result = TAP::Parser::Result->new( $token );
```

DESCRIPTION

This is a simple base class used by *TAP::Parser* to store objects that represent the current bit of test output data from TAP (usually a single line). Unless you're subclassing, you probably won't need to use this module directly.

METHODS

new

```
  # see TAP::Parser::ResultFactory for preferred usage

  # to use directly:
  my $result = TAP::Parser::Result->new($token);
```

Returns an instance the appropriate class for the test token passed in.

Boolean methods

The following methods all return a boolean value and are to be overridden in the appropriate subclass.

* `is_plan`

  Indicates whether or not this is the test plan line.

  ```
  1..3
  ```

* `is_pragma`

  Indicates whether or not this is a pragma line.

  ```
  pragma +strict
  ```

* `is_test`

  Indicates whether or not this is a test line.

  ```
  ok 1 Is OK!
  ```

* `is_comment`

  Indicates whether or not this is a comment.

  ```
  # this is a comment
  ```

* `is_bailout`

Indicates whether or not this is bailout line.

```
Bail out! We're out of dilithium crystals.
```

* is_version

  Indicates whether or not this is a TAP version line.

```
TAP version 4
```

* is_unknown

  Indicates whether or not the current line could be parsed.

```
... this line is junk ...
```

* is_yaml

  Indicates whether or not this is a YAML chunk.

raw

```
print $result->raw;
```

Returns the original line of text which was parsed.

type

```
my $type = $result->type;
```

Returns the "type" of a token, such as `comment` or `test`.

as_string

```
print $result->as_string;
```

Prints a string representation of the token. This might not be the exact output, however. Tests will have test numbers added if not present, TODO and SKIP directives will be capitalized and, in general, things will be cleaned up. If you need the original text for the token, see the `raw` method.

is_ok

```
if ( $result->is_ok ) { ... }
```

Reports whether or not a given result has passed. Anything which is **not** a test result returns true. This is merely provided as a convenient shortcut.

passed

Deprecated. Please use `is_ok` instead.

has_directive

```
if ( $result->has_directive ) {
    ...
}
```

Indicates whether or not the given result has a TODO or SKIP directive.

has_todo

```
if ( $result->has_todo ) {
    ...
}
```

Indicates whether or not the given result has a TODO directive.

has_skip

```
if ( $result->has_skip ) {
    ...
}
```

Indicates whether or not the given result has a SKIP directive.

set_directive

Set the directive associated with this token. Used internally to fake TODO tests.

SUBCLASSING

Please see *"SUBCLASSING" in TAP::Parser* for a subclassing overview.

Remember: if you want your subclass to be automatically used by the parser, you'll have to register it with *"register_type" in TAP::Parser::ResultFactory*.

If you're creating a completely new result *type*, you'll probably need to subclass *TAP::Parser::Grammar* too, or else it'll never get used.

Example

```
package MyResult;

use strict;
use vars '@ISA';

@ISA = 'TAP::Parser::Result';

# register with the factory:
TAP::Parser::ResultFactory->register_type( 'my_type' => __PACKAGE__ );

sub as_string { 'My results all look the same' }
```

SEE ALSO

TAP::Object, *TAP::Parser*, *TAP::Parser::ResultFactory*, *TAP::Parser::Result::Bailout*, *TAP::Parser::Result::Comment*, *TAP::Parser::Result::Plan*, *TAP::Parser::Result::Pragma*, *TAP::Parser::Result::Test*, *TAP::Parser::Result::Unknown*, *TAP::Parser::Result::Version*, *TAP::Parser::Result::YAML*,