NAME

"IO::Async::Future" – use Future with IO::Async

SYNOPSIS

```
 use IO::Async::Loop;

 my $loop = IO::Async::Loop->new;

 my $future = $loop->new_future;

 $loop->watch_time( after => 3, code => sub { $future->done( "Done" ) } );

 print $future->get, "\n";
```

DESCRIPTION

This subclass of Future stores a reference to the IO::Async::Loop instance that created it, allowing the `await` method to block until the Future is ready. These objects should not be constructed directly; instead the `new_future` method on the containing Loop should be used.

For a full description on how to use Futures, see the Future documentation.

CONSTRUCTORS

New `IO::Async::Future` objects should be constructed by using the following methods on the `Loop`. For more detail see the IO::Async::Loop documentation.

```
 $future = $loop->new_future
```

Returns a new pending Future.

```
 $future = $loop->delay_future( %args )
```

Returns a new Future that will become done at a given time.

```
 $future = $loop->timeout_future( %args )
```

Returns a new Future that will become failed at a given time.

METHODS

loop

```
 $loop = $future->loop
```

Returns the underlying IO::Async::Loop object.

done_later

```
 $future->done_later( @result )
```

A shortcut to calling the `done` method in a `later` idle watch on the underlying Loop object. Ensures that a returned Future object is not ready immediately, but will wait for the next IO round.

Like `done`, returns `$future` itself to allow easy chaining.

fail_later

```
 $future->fail_later( $exception, @details )
```

A shortcut to calling the `fail` method in a `later` idle watch on the underlying Loop object. Ensures that a returned Future object is not ready immediately, but will wait for the next IO round.

Like `fail`, returns `$future` itself to allow easy chaining.

AUTHOR

Paul Evans <leonerd@leonerd.org.uk>