series()

Combines task functions and/or composed operations into larger operations that will be executed one after another, in sequential order. There are no imposed limits on the nesting depth of composed operations using series() and parallel().

Usage

```
const { series } = require('gulp');

function javascript(cb) {
  // body omitted
  cb();
}

function css(cb) {
  // body omitted
  cb();
}

exports.build = series(javascript, css);
```

Signature

```
series(...tasks)
```

Parameters

parameter	type	note
tasks (required)	function string	Any number of task functions can be passed as individual arguments. Strings can be used if you've registered tasks previously, but this is not recommended.

Returns

A composed operation to be registered as a task or nested within other series and/or parallel compositions.

When the composed operation is executed, all tasks will be run sequentially. If an error occurs in one task, no subsequent tasks will be run.

Errors

When no tasks are passed, throws an error with the message, "One or more tasks should be combined using series or parallel".

When invalid tasks or unregistered tasks are passed, throws an error with the message, "Task never defined".

Forward references

A forward reference is when you compose tasks, using string references, that haven't been registered yet. This was a common practice in older versions, but this feature was removed to achieve faster task runtime and promote the use of named functions.

In newer versions, you'll get an error, with the message "Task never defined", if you try to use forward references. You may experience this when trying to use exports for your task registration *and*composing tasks by string. In this situation, use named functions instead of string references.

During migration, you may need to use the forward reference registry. This will add an extra closure to every task reference and dramatically slow down your build. Don't rely on this fix for very long.

Avoid duplicating tasks

When a composed operation is run, each task will be executed every time it was supplied.

A clean task referenced in two different compositions would be run twice and lead to undesired results. Instead, refactor the clean task to be specified in the final composition.

If you have code like this:

```
// This is INCORRECT
const { series, parallel } = require('gulp');

const clean = function(cb) {
  // body omitted
  cb();
};

const css = series(clean, function(cb) {
  // body omitted
  cb();
});

const javascript = series(clean, function(cb) {
  // body omitted
  cb();
});

exports.build = parallel(css, javascript);
```

Migrate to this:

```
const { series, parallel } = require('gulp');

function clean(cb) {
  // body omitted
  cb();
}

function css(cb) {
  // body omitted
  cb();
}

function javascript(cb) {
  // body omitted
  cb();
}

exports.build = series(clean, parallel(css, javascript));
```