# Gulp Streaming build tasks

**START HERE**

**`npm install --save-dev gulp`**
Add gulp to a project

**`npm install --global gulp-cli`**
Get the gulp command

**`gulp`**
Run the default task

**`gulp <task>`**
Run one task

**`gulp --tasks`**
List tasks

**`gulp.src(<glob>)`**
Read files

**`.pipe(<plugin>())`**
Transform them

**`gulp.dest(<dir>)`**
Write the result

**src()** glob in, files out → pipe → **Plugins** transform in memory → pipe → **dest()** write to disk

## CONCEPTS

A task is an async function in gulpfile.js; export it to make it runnable from the command line.

Files travel as Vinyl objects (path plus contents); plugins take and return them.

A task must signal completion: return a stream, promise or child process, take a done callback, or be async. Otherwise gulp reports it never finished.

## INSTALL

**`npm install --save-dev gulp`**
Add gulp to the project

**`npm install --global gulp-cli`**
Install the gulp command globally

**`npx gulp <task>`**
Run the local CLI without a global install

**`gulp --version`**
Print CLI and local gulp versions

**`gulp --help`**
Show all CLI flags

## RUN

**`gulp`**
Run the default task

**`gulp <task>`**
Run one exported task

**`gulp <task1> <task2>`**
Run several tasks in parallel
*Parallel is the CLI default; add --series to keep the given order.*

**`gulp --series <task1> <task2>`**
Run CLI tasks one after another

**`gulp --continue <task>`**
Keep going after a task fails

**`gulp -f <path> <task>`**
Use another gulpfile and set cwd to it

**`gulp --cwd <dir> <task>`**
Look for the gulpfile from another directory

**`gulp --preload <module> <task>`**
Load a module before the gulpfile

**`gulp --silent <task>`**
Suppress gulp logging

**`gulp -LLLL <task>`**
Most verbose logging (-L least)

## RUN (cont.)

**`gulp --no-color <task>`**
Disable coloured output

## INSPECT TASKS

**`gulp --tasks`**
Print the task tree (also -T)

**`gulp --tasks-simple`**
Plain list of task names

**`gulp --tasks-json`**
Task tree as JSON

**`gulp --tasks --depth <n>`**
Limit task tree depth

**`gulp --tasks --sort-tasks`**
Sort top-level tasks

**`gulp --tasks --compact-tasks`**
Show only top tasks and children

## DEFINE TASKS

**`exports.<name> = <fn>`**
Make a task runnable from the CLI

**`exports.default = <fn>`**
Task run by plain gulp

**`gulp.task('<name>', <fn>)`**
Register a task by string name
*Works, but exporting functions is the recommended style.*

**`gulp.series(<a>, <b>)`**
Run tasks in order

**`gulp.parallel(<a>, <b>)`**
Run tasks concurrently

**`async function <name>() { }`**
Task using async/await

**`function <name>(cb) { cb() }`**
Task using a done callback

**`cb(err)`**
Fail a callback task

**`<fn>.displayName = '<name>'`**
Name an anonymous function task

**`<fn>.description = '<text>'`**
Text shown by --tasks

**`gulp.series(<a>, gulp.parallel(<b>, <c>))`**
Nest compositions freely

**`gulp.tree()`**
Task tree object, as in --tasks-json

## FILES AND STREAMS

**`gulp.src('<glob>')`**
Read files matching a glob

## FILES AND STREAMS (cont.)

**`gulp.src(['<glob>', '!<glob>'])`**
Include and exclude globs

**`gulp.dest('<dir>')`**
Write files to a directory

**`.pipe(<transform>)`**
Pass the stream through a plugin

**`gulp.src('<glob>', { base: '<dir>' })`**
Keep folder structure from a base

**`gulp.src('<glob>', { encoding: false })`**
Read as binary (images, fonts)

**`gulp.src('<glob>', { buffer: false })`**
Stream contents for big files

**`gulp.src('<glob>', { allowEmpty: true })`**
No error when a glob matches nothing

**`gulp.src('<glob>', { ignore: '<glob>' })`**
Exclude globs via an option

**`gulp.src('<glob>', { dot: true })`**
Match dotfiles too

**`gulp.src('<glob>', { sourcemaps: true })`**
Load and track sourcemaps

**`gulp.dest('<dir>', { sourcemaps: '.' })`**
Write external sourcemap files

**`gulp.src('<glob>', { since: gulp.lastRun('<task>') })`**
Only files changed since the last run

### Glob patterns

| Pattern | Matches |
|---|---|
| `*` | Any name within one directory |
| `**` | Any depth of directories |
| `!<glob>` | Exclude matches |
| `{a,b}` | Either alternative |

## WATCH

**`gulp.watch('<glob>', <task>)`**
Run a task when files change

**`gulp.watch('<glob>', gulp.series(<a>, <b>))`**
Watch and run a composition

**`gulp.watch('<glob>').on('change', <fn>)`**
React to a single event

**`gulp.watch('<glob>', { ignoreInitial: false }, <task>)`**
Also run once at startup

## WATCH (cont.)

**`gulp.watch('<glob>', { events: ['add'] }, <task>)`**
Limit to chosen events

**`gulp.watch('<glob>', { delay: 500 }, <task>)`**
Debounce changes (ms)

**`gulp.watch('<glob>', { queue: false }, <task>)`**
Allow overlapping runs

**`gulp.watch('<glob>', { usePolling: true }, <task>)`**
Poll for changes (network drives)

watch() ignores changes while its task runs and queues one more run; set queue to false to disable that.

## RECIPES

### Minimal gulpfile.js

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

function copy() {
  return src('src/**/*').pipe(dest('dist'));
}

exports.build = series(copy);
exports.default = exports.build;
```

### Clean then build

```
const { rm } = require('node:fs/promises');

async function clean() {
  await rm('dist', { recursive: true, force: true });
}

exports.build = series(clean, parallel(styles, scripts));
```

### Dev loop: build once, then watch

```
function watchFiles() {
  watch('src/**/*.js', scripts);
  watch('src/**/*.css', styles);
}

exports.dev = series(exports.build, watchFiles);
```

### Transform with a plugin and sourcemaps

```
// npm i -D gulp-uglify
const uglify = require('gulp-uglify');

function scripts() {
  return src('src/**/*.js', { sourcemaps: true })
    .pipe(uglify())
    .pipe(dest('dist', { sourcemaps: '.' }));
}
```