Gulp Streaming build tasks

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

INSTALL (cont.)

```
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

```
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)

INSPECT TASKS

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

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

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

```
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

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

FILES AND STREAMS

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

```
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>', { allowEmpty: true })
```
No error when a glob matches nothing

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

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

FILES AND STREAMS (cont.)

```
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

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