# Gulp Streaming build tasks

## CONCEPTS

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

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

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

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

## 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.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)

## WATCH

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

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