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