# Passthrough Arguments

If you have a shortcut for running a specific combination of commands through concurrently, you might need at some point to pass additional arguments/flags to some of these.

For example, imagine you have in your package.json file scripts like this:

{

// ...

"scripts": {

"build:client": "tsc -p client",

"build:server": "tsc -p server",

"build": "concurrently npm:build:client npm:build:server"

}

}

If you wanted to run only either build:server or build:client with an additional --noEmit flag, you can do so with npm run build:server -- --watch, for example.

However, if you want to do that while using concurrently, as `npm run dev -- --noEmit` for example, you might find that concurrently actually parses `--watch` as its own flag, which does nothing, because it doesn't exist.

To solve this, you can set the --passthrough-arguments/-P flag, which instructs concurrently to take everything after a -- as additional arguments that are passed through to the input commands via a few placeholder styles:

## Single argument

We can modify the original build script to pass a single additional argument/flag to a script by using a 1-indexed {number} placeholder to the command you want it to apply to:

{

// ...

"scripts": {

// ...

"build": "concurrently -P 'npm:build:client -- {1}' npm:build:server --",

"typecheck": "npm run build -- --noEmit"

}

}

With this, running npm run typecheck will pass --noEmit only to npm run build:client.

## All arguments

In the original build example script, you're more likely to want to pass every additional argument/flag to your commands. This can be done with the {@} placeholder.

{

// ...

"scripts": {

// ...

"build": "concurrently -P 'npm:build:client -- {@}' 'npm:build:server -- {@}' --",

"typecheck": "npm run build -- --watch --noEmit"

}

}

In the above example, both --watch and --noEmit are passed to each command.

## All arguments, combined

If for some reason you wish to combine all additional arguments into a single one, you can do that with the {\*} placeholder, which wraps the arguments in quotes.

{

// ...

"scripts": {

// ...

"build": "concurrently -P 'npm:build:client -- --outDir {\*}/client' 'npm:build:server -- --outDir {\*}/server' -- $(date)"

}

}

In the above example, the output of the date command, which looks like Sun 1 Sep 2024 23:50:00 AEST will be passed as a single string to the --outDir parameter of both commands.