Working with Files

The src() and dest() methods are exposed by gulp to interact with files on your computer.
src() is given a glob to read from the file system and produces a Node stream. It locates all matching files and reads them into memory to pass through the stream.
The stream produced by src() should be returned from a task to signal async completion, as mentioned in Creating Tasks.

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

exports.default = function() {
  return src('src/*.js')
    .pipe(dest('output/'));
}
```

The main API of a stream is the .pipe() method for chaining Transform or Writable streams.

```
const { src, dest } = require('gulp');
const babel = require('gulp-babel');

exports.default = function() {
  return src('src/*.js')
    .pipe(babel())
    .pipe(dest('output/'));
}
```

dest() is given an output directory string and also produces a Node stream which is generally used as a terminator stream. When it receives a file passed through the pipeline, it writes the contents and other details out to the filesystem at a given directory.
The symlink() method is also available and operates like dest(), but creates links instead of files (see symlink() for details).
Most often plugins will be placed between src() and dest() using the .pipe() method and will transform the files within the stream.

Adding files to the stream

src() can also be placed in the middle of a pipeline to add files to the stream based on the given globs. The additional files will only be available to transformations later in the stream. If globs overlap, the files will be added again.
This can be useful for transpiling some files before adding plain JavaScript files to the pipeline and uglifying everything.

```
const { src, dest } = require('gulp');
const babel = require('gulp-babel');
const uglify = require('gulp-uglify');

exports.default = function() {
  return src('src/*.js')
    .pipe(babel())
    .pipe(src('vendor/*.js'))
    .pipe(uglify())
    .pipe(dest('output/'));
}
```

Output in phases

dest() can be used in the middle of a pipeline to write intermediate states to the filesystem. When a file is received, the current state is written out to the filesystem, the path is updated to represent the new location of the output file, then that file continues down the pipeline.
This feature can be useful to create unminified and minified files with the same pipeline.

```
const { src, dest } = require('gulp');
const babel = require('gulp-babel');
const uglify = require('gulp-uglify');
const rename = require('gulp-rename');

exports.default = function() {
  return src('src/*.js')
    .pipe(babel())
    .pipe(src('vendor/*.js'))
    .pipe(dest('output/'))
    .pipe(uglify())
    .pipe(rename({ extname: '.min.js' }))
    .pipe(dest('output/'));
}
```

Modes: streaming, buffered, and empty

src() can operate in three modes: buffering, streaming, and empty. These are configured with the buffer and read options on src().

Buffering mode is the default and loads the file contents into memory. Plugins usually operate in buffering mode and many don't support streaming mode.

Streaming mode exists mainly to operate on large files that can't fit in memory, like giant images or movies. The contents are streamed from the filesystem in small chunks instead of loaded all at once. If you need to use streaming mode, look for a plugin that supports it or write your own.

Empty mode contains no contents and is useful when only working with file metadata.