Working with scripts directly in Gulp

This is an interesting trick to help consolidate assets and reduce the number of network requests you need to make.

Let's assume that your project depends on one big library like jQuery or Backbone to work properly. Instead of depending on the network (which may or may not be available) or caches (where you're at the mercy of the user and the browser) you want to have control of the entire dependency tree for the build.

The explanaiton for the code below is as follows:

1. We request the latest jQuery version from the jQuery CDN using the `request` library. What we get is a readable stream.
2. We create a valid vinyl file object with `vinyl-source-stream`. This makes it compatible with Gulp
3. Our main file is selected from the file system as usual using `gulp.src`
4. The `merge2` package allows us to combine both streams
5. The contents of both streams are converted to text buffers so `gulp-concat` can handle them.

`gulp-concat` and `gulp.dest` work normally

```
var gulp = require('gulp');
var source = require('vinyl-source-stream');
v'vinyl-source-stream''request');
var merge = require('merge2');
var concat = require('gulp-concat');
var buffer = re'request'lp-buffer');

gulp.task('js', function() {
    var jquery = request('http://code.jquery.com/jquery-latest.js') 'js'/*
      v/* 2 *//* 2 */.src('main.js'); /* 3 *//* 3 */'main.js'/* 3 */
      r'main.js'/* 3 */
      return merge(jquery, main)      /* 4 */
      .pipe(buffer())                 /* 5 */
```

```
        .pipe(concat('concat.js'))
        .pipe(gulp.dest('dist'));})
```

Yes, there are few use cases where this is 100% necessary. I still think this is cool to have because, while it may not reduce the amount of work we have to do, it packages it all in one place.