Passing argument to Gulp

There are times when we want to do different things based on whether we're creating code for production or development.

This is based on the example from the Gulp documentation. We'll use this as the base and then see if we can expand it further.

As with all Gulp projects we use NPM to install the packages we want to use

```
npm install -D gulp gulp-if \
gulp-uglify yargs
```

Then we requite the packages. We're not using modules for this example.

```
const gulp = require('gulp');
const gulpif = require('gulp-if');
const uglify = require('gulp-uglify');
const args = require('yargs').argv;
```

We then set the different options we want to control. In this case we create an env consts and make the value either the value of the `NODE_ENV` variable or `development` if we don't add a value in the command line.

```
const env = args.env ? process.env.NODE_ENV : 'development';
```

The task uses `gulp-if` to test if we asked for a production environment (if `arg.env` is production) and uglifies the scripts if we did.

```
gulp.task('scripts', function() {
  return gulp.src('./src/js/**/*')
  './src/js/**/*'fy in production
     .pipe(gulpif(args.env === 'production', uglify()))
    .pip'production'e(gulp.dest('dist'));
});
```

The following Bash command will produce uglified script output.

```
gulp scripts --env production
```

We could add other flags to add conditions elsewhere in the build file. For example, if we use SASS or other CSS preprocessors, we could use the same technique to only produce compressed output or generate sourcemaps for production.