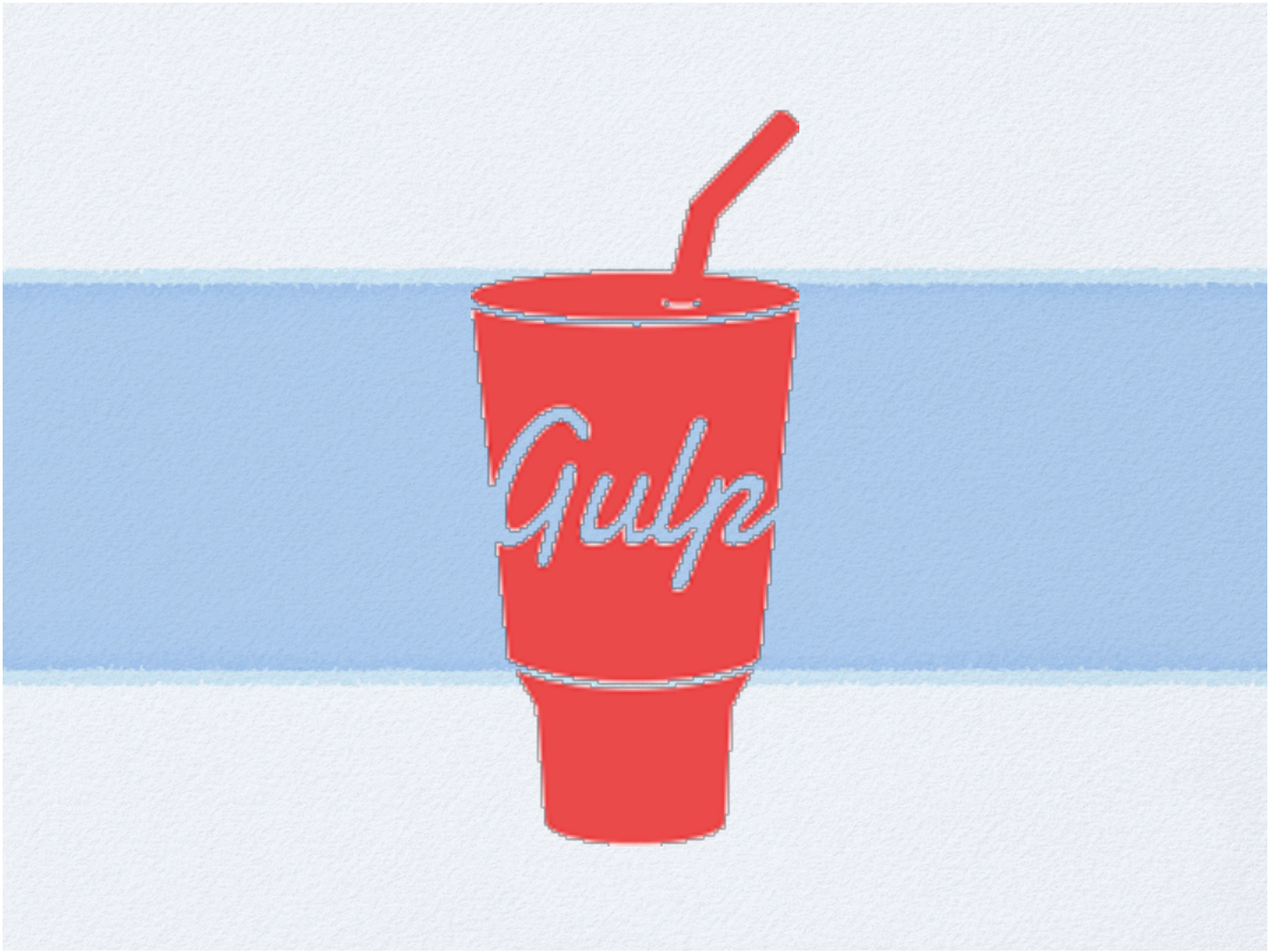

gulp

Overview

- JavaScript-based build tool for executing many web development tasks
  - competes with Grunt
  - gulp generally runs faster due to use of streams
    - uses streams instead of temporary files to make data available to subsequent steps
    - allows subsequent steps to begin earlier, as soon as data is available in stream instead of waiting for a file to be completely written
- Runs on Node.js
  - so must install that first

there are issues with Node.js stdout output in Cygwin, so some gulp output may not be visible there

- Tasks are added via plugins
- There are MANY plugins! ... 1,457 as of 3/8/15

```
npm search gulpplugin | wc -l
```

- http://gulpjs.com

Task Examples

- **Beautify** JavaScript
- **Validate**/lint .html, .css, .js and .json files
- **Compile** many kinds of HTML templates
- Compile CoffeeScript to JavaScript
- Compile Compass/LESS/SASS/Stylus to CSS
- **Run** many kinds of **tests** (Jasmine, Mocha, Nodeunit, QUnit, Selenium, Vows, ...)
  - can run browser tests using PhantomJS
- **Generate documentation** from code
- **Concatenate** and **minimize** .css and .js files
- **Copy** files to a server using ftp, scp or sftp
- **Deploy** web apps
- **Serve** static **files** via HTTP
- Run **Git** and Subversion commands
- Perform JSON **schema validation**
- Execute **shell commands**
- **Watch** for file changes and run tasks
  - including reloading affected browser page
- Perform **spell checking**
- Release new versions of NPM projects
- Display **Growl** notifications
- Play **sounds**
- Compile **ES6** JavaScript to ES3 JavaScript (using Babel or Traceur)
- Compile TypeScript to JavaScript
- **Zip** and unzip files

Installing

- **`npm install -g gulp`** for command-line access
  - for 4.0,
    ```
    npm uninstall -g gulp
    npm install -g gulpjs/gulp-cli#4.0
    ```
- cd to project directory
- if no **`package.json`** exists yet, create with **`npm init`**
  - will ask lots of questions; can accept the default for most
- **`npm install gulp --save-dev`** for actually running
  - for 4.0
    ```
    npm uninstall gulp
    npm install gulpjs/gulp.git#4.0 --save-dev
    ```
    `--save-dev` option is explained later
  - installs in **`node_modules`** subdirectory
- Create **`gulpfile.js`**
  - see example later

In some environments, Node.js can only find globally installed modules if the **`NODE_PATH`** environment variable is set to tell it where to look.
Running "**`npm root -g`**" will output the directory where globally installed modules reside.
In Windows,
`set NODE_PATH=`*`root-dir`*
In *nix,
`export NODE_PATH=`*`root-dir`*

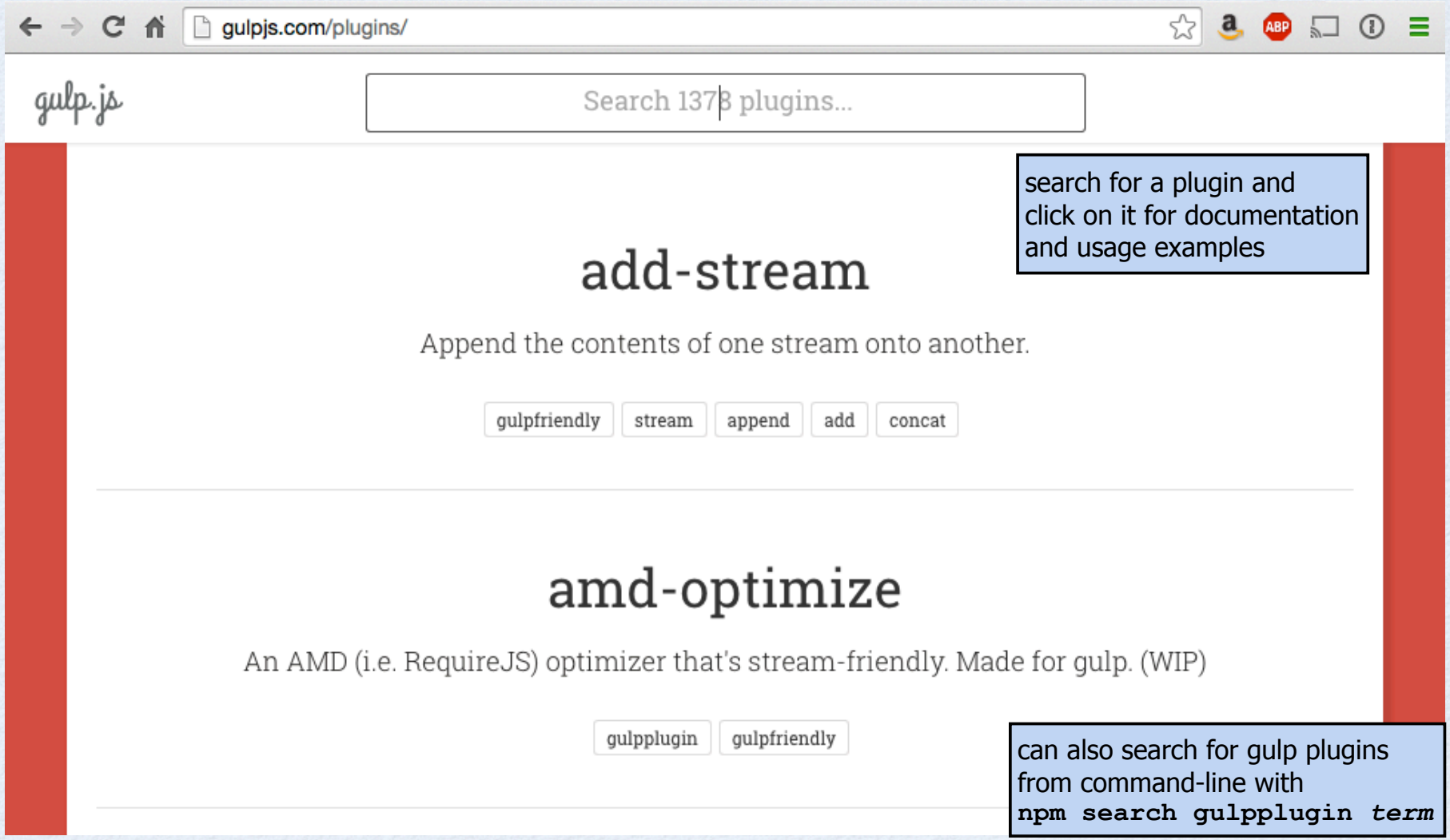
The remaining slides cover **gulp 4**, not gulp 3.

Running

- cd to project directory where gulp has already been installed
- **`gulp --help`** or **`-h`**
  - lists available options with a description of each
- **`gulp --version`** or **`-v`** - prints global and local version
- **`gulp --tasks`** or **`-T`** - prints tasks dependency tree
- **`gulp --tasks-simple`** - prints list of tasks without showing dependencies
- **`gulp --verify`** - checks **`package.json`** for use of blacklisted plugins
- **`gulp options task1 task2 ...`**
  - runs all tasks specified in parallel
  - if no tasks are specified, the default task is run
    - if no default is defined, an error message is displayed and gulp aborts
  - most tasks run to completion and **`gulp`** terminates
  - some run until killed (ctrl-c), such as **`connect`** and **`watch`**

we'll see how to define a default task later

gulp Plugins Page



search for a plugin and click on it for documentation and usage examples

can also search for gulp plugins from command-line with **`npm search gulpplugin term`**

Recommended Plugins ...

plugins that are published to npm with the "gulpplugin" keyword are automatically cataloged at http://gulpjs.com/plugins

- **del** - deletes specified directories and files
- **gulp-babel** - compiles ES6 files to ES5
- **gulp-changed** - only processes src files that are newer than dest file
- **gulp-concat** - concatenates CSS and JavaScript files
- **gulp-csslint** - validates CSS files
- **gulp-eslint** - validates JavaScript files
- **gulp-jasmine** - runs Jasmine tests
- **gulp-jshint** - validates JavaScript files
- **gulp-less** - compiles LESS files to CSS
- **gulp-livereload** - reloads browser when `livereload` is called

… Recommended Plugins

- **gulp-plumber** - allows to continue running after errors
  - not needed for most tasks in Gulp 4, but needed with Jasmine
- **gulp-sourcemaps** - generates sourcemaps that allow debugging in files that are compiled to the JavaScript that runs in the browser
- **gulp-uglify** - minimizes JavaScript files
- **gulp-usemin** - "replaces references to non-optimized scripts or stylesheets into a set of HTML files"
- **gulp-watch** - watches files for changes and runs specified tasks when they do

Installing a Plugin

- See searchable list at http://gulpjs.com/plugins
- **`npm install plugin-name --save-dev`**
  - installs in local **`node_modules`** directory
    - can also install globally with **`-g`** so multiple projects can share the plugin
  - **`--save-dev`** option causes **`package.json`** to be modified so **`devDependencies`** property contains a reference to the plugin
  - can delete **`node_modules`** directory and get all plugins back by running **`npm install`**
- Edit gulpfile.js
  - add a **`require`** for each plugin, and one for gulp itself

```
var gulp = require('gulp');
var name = require('plugin-name');
...
```

  - use the plugin from one or more tasks

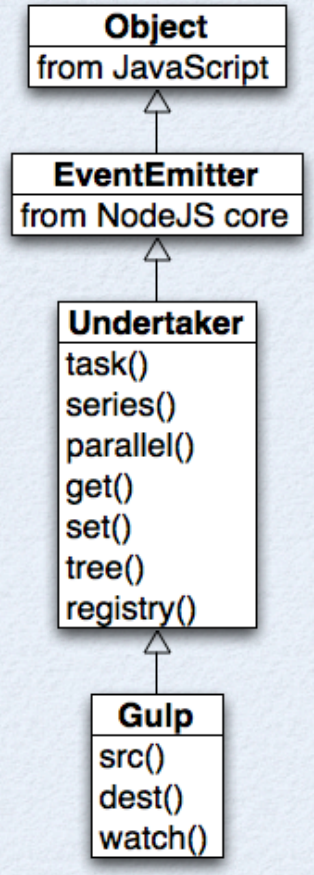
Automatically Requiring Plugins

- When a new plugin is installed, **`gulpfile.js`** must be modified to require and use it
  - **`var *name* = require('*plugin-name*');`**
- Can automate with **gulp-load-plugins** module
  - returns an object whose properties are all gulp plugin dependency names found in **`package.json`**
  - to install, **`npm install gulp-load-plugins --save-dev`**
  - in **`gulpfile.js`**, replace all gulp plugin requires with:

```
var gulp = require('gulp');
var pi = require('gulp-load-plugins')();
```

  - reference plugins in tasks with **`pi.*name*`**
  - lazily loads plugins - not loaded until their first use

gulp 4 Methods ...

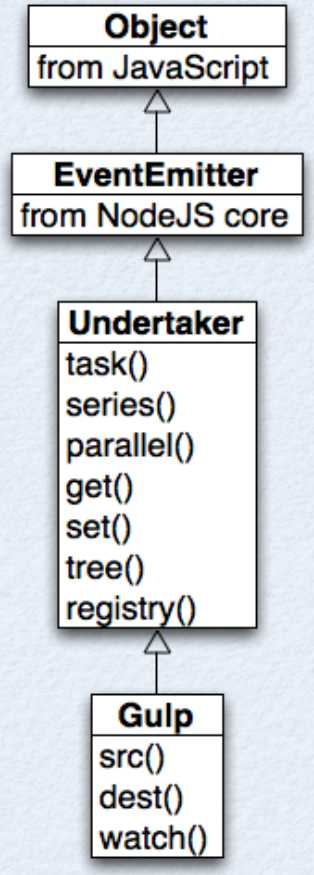


- `Gulp` methods
  - `src(glob, opt)` - specifies input files to be processed
  - `dest(outFolder, opt)` - specifies output file or directory
  - `watch(glob, opt, task)` - watches files (add, modify, delete) and runs given task
    - task will fire twice if editor modifies file twice
    - Vim does this by default as part of its backup strategy
      - to prevent it, add "set nobackup" to .vimrc
      - see http://stackoverflow.com/questions/21608480/gulp-js-watch-task-runs-twice-when-saving-files and http://stackoverflow.com/questions/607435/why-does-vim-save-files-with-a-extension

glob arguments can be a string or array of strings containing wildcard characters

none of the supported **options** are commonly used

… gulp 4 Methods



- `Undertaker` methods
  - `task(name, fn)` - defines a new task
    - `fn` can be a named function defined elsewhere or an anonymous function defined here
    - if `fn` is omitted, the function previously registered with the name is returned
  - `series(tasks)` - executes tasks in series and returns a task representing the set
  - `parallel(tasks)` - executes tasks in parallel and returns a task representing the set

    use `series` and `parallel` methods to run more than one task

    `tasks` can be task name strings or functions

    `series` and `parallel` methods return a function that can be passed to the `task` method
  - * `get(name)` - returns task function for given task name
  - * `set(name, fn)` - sets/changes function for given task name
    - * if task is already defined, it is replaced
  - * `tree(opts)` - returns array of defined task names
    - * if `opts` sets `deep` to `true`, get array of objects that describe task dependencies
  - * `registry(newRegistry)` - gets/sets map of task names to functions `{_tasks: map}`

existence of `Undertaker` class is an implementation detail

* methods are not typically used directly

Defining Tasks

- Syntax

```
gulp.task(name, function () { ... });
gulp.task(name, gulp.series(...));
gulp.task(name, gulp.parallel(...));
```

- Simple example

```
var gulp = require('gulp');

gulp.task('hello', function () {
  console.log('Hello, World!');
});
```

  - run with "`gulp hello`"

Serving Files

- Can use **connect**, **http**, and **serveStatic** node modules to serve static files such as HTML, CSS, JavaScript, and images

```
var connect = require('connect');
var http = require('http');
var serverStatic = require('serveStatic');
...
  gulp.task('connect', function () {
    var app = connect();
    app.use(serveStatic(__dirname));
    http.createServer(app).listen(port);
  });
```

__dirname is a Node.js variable that holds the path to the current directory

- See **gulpfile.js** example ahead

Watching For Changes

- Runs specified tasks when new files are created or existing files are modified
- Configure with files paths and array of task names or a function to run

```
gulp.task('watch', function () {
  livereload.listen();
  gulp.watch(paths.html, 'html');
  gulp.watch(paths.less, gulp.series('less', 'csslint'));
  gulp.watch(paths.jsWithTests,
    gulp.series('eslint', 'jshint', 'transpile-dev'));
});
```

properties in paths object are glob pattern strings or arrays of them

with livereload, CSS changes are injected without reloading the page

- Need to restart gulp after modifying **`gulpfile.js`**

Live Reload

- Causes browser to reload after certain files are modified
- Can use **gulp-livereload** plugin
  - works best with Chrome
  - must install "livereload" Chrome extension
    - see link at https://www.npmjs.com/package/gulp-livereload
- Steps to use
  - add livereload `script` tag to main HTML file

```
<script src="http://localhost:35729/livereload.js"></script>
```

  - call `livereload.listen()` in `watch` task
  - call `livereload()` after every file change that should trigger a reload
    - ex. HTML, CSS, or JavaScript

gulpfile.js Example ...

```
var connect = require('connect');
var del = require('del');
var gulp = require('gulp');
var http = require('http');
var pi = require('gulp-load-plugins')();
var serveStatic = require('server-static');

var paths = {
  build: 'build',
  css: 'build/**/*.css',
  html: ['index.html', 'src/**/*.html'],
  js: ['src/**/*.js'],
  jsPlusTests: ['src/**/*.js', 'test/**/*.js'],
  less: 'src/**/*.less',
  test: 'build/**/*-test.js'
};

gulp.task('hello', function () {
  console.log('Hello, World!');
});
```

```
gulp.task('clean', function (cb) {
  del(paths.build, cb);
});

gulp.task('connect', function () {
  var app = connect();
  app.use(serveStatic(__dirname));
  http.createServer(app).listen(1919);
});

gulp.task('csslint', function () {
  return gulp.src(paths.css).
    pipe(pi.csslint({
      ids: false
    })).
    pipe(pi.csslint.reporter());
});
```

... gulpfile Example ...

```
gulp.task('eslint', function () {
  return gulp.src(paths.jsPlusTests).
    pipe(pi.changed(paths.build)).
    pipe(pi.eslint({
      envs: ['browser', 'es6', 'node'],
      rules: {
        curly: [2, 'multi-line'],
        indent: [2, 2]
      }
    })).
    pipe(pi.eslint.format());
});

gulp.task('html', function () {
  gulp.src(paths.html).
    pipe(pi.livereload());
});
```

first 2 means treat violations as errors

```
gulp.task('jshint', function () {
  return gulp.src(paths.jsPlusTests).
    pipe(pi.changed(paths.build)).
    pipe(pi.jshint()).
    pipe(pi.jshint.reporter('default'));
});

gulp.task('less', function () {
  return gulp.src(paths.less).
    pipe(pi.less()).
    pipe(pi.changed(paths.build)).
    pipe(gulp.dest(paths.build)).
    pipe(pi.livereload());
});
```

... gulpfile Example ...

```
gulp.task('transpile-dev', function () {
  return gulp.src(paths.jsPlusTests).
    pipe(pi.changed(paths.build)).
    pipe(pi.sourcemaps.init()).
    pipe(pi.babel()).
    pipe(pi.sourcemaps.write('.')).
    pipe(gulp.dest(paths.build)).
    pipe(pi.livereload());
});

gulp.task('transpile-prod', function () {
  return gulp.src(paths.js).
    pipe(pi.sourcemaps.init()).
    pipe(pi.babel()).
    pipe(pi.concat('all.js')).
    pipe(pi.uglify()).
    pipe(pi.sourcemaps.write('.')).
    pipe(gulp.dest(paths.build));
});
```

```
gulp.task('test',
  gulp.series('transpile-dev',
    function () {
      return gulp.src(paths.test).
        pipe(pi.plumber()).
        pipe(pi.jasmine());
    }));
```

... gulpfile Example

```
gulp.task('watch', function () {
  pi.livereload.listen();
  gulp.watch(paths.html, 'html');
  gulp.watch(paths.less, gulp.series('less', 'csslint'));
  gulp.watch(paths.jsPlusTests,
    gulp.series('eslint', 'jshint', 'transpile-dev'));
});

gulp.task('build-dev', gulp.parallel('less', 'transpile-dev'));
gulp.task('build-prod', gulp.parallel('less', 'transpile-prod'));

gulp.task('default',
  gulp.series('build-dev', gulp.parallel('connect', 'watch')));
```

Demo ...

- **`cd javascript-labs/gulp/gulp-demo`**
- **`gulp clean`**
  - deletes build directory created by previously run commands
- **`gulp less`**
  - compiles all **`.less`** files in **`src`** directory to **`.css`** files in **`build`**
    - generates **`build/demo.css`**
- **`gulp csslint`**
  - checks for issues in **`.css`** files in **`src`** directory
    - try changing **`red`** to **`rod`** in **`src/demo.css`**
- **`gulp transpile-dev`**
  - generates ES5 **`.js`** files in **`build`** directory from ES6 **`.js`** files in **`src`** directory
- **`gulp jshint`**
  - checks for issues in **`.js`** files in **`src`** directory
    - try removing a semicolon in **`src/demo.js`**

… Demo

- **`gulp test`**
  - runs Jasmine tests below **`test`** directory
    - change `toBe('Foo')` to `toBe('Bar')` in **`test/str-util-test.js`** and run this
- **`gulp`**
  - starts local HTTP server then
    watches for changes to files with several extensions,
    runs the appropriate tasks on them,
    and reloads browser window
    - browse **`localhost:1919`**
    - modify the following files, observe terminal output, and look for changes in reloaded browser window
    - **`index.html`** - reloads — change body content
    - **`src/demo.less`** - generates CSS and lints — change `@title-color`
    - **`src/demo.js`** - transpiles and lint — change value of **`title.textContent`**

Lab ...

- If on Windows, set NODE_PATH environment variable
  - see box on slide 8-4
- Install gulp 4
- Configure Gulp for the Todo application in **`lab7`**
  - rename **`package.json`** to **`package-solution.json`**
  - create **`package.json`** by running "**`npm init`**"
  - delete the **`node_modules`** directory
  - install gulp version 4 - see slide 4
  - install the plugins to be used - see slide 9
    - install these: **`gulp-csslint`**, **`gulp-jshint`**, **`gulp-livereload`**, **`gulp-load-plugins`**, **`gulp-watch`**
    - install these: **`body-parser`**, **`express`**

... Lab

- rename **`gulpfile.js`** to **`gulpfile-solution.js`**
- create **`gulpfile.js`** from scratch with tasks for **`csslint`**, **`jshint`** and **`watch`** (use **`livereload`**)
  - register default task to run all of these in the order above
  - use **`jshint`** to check **`gulpfile.js`**
- verify that **`index.html`** contains this **`script`** tag
  - `<script src="http://localhost:35729/livereload.js"></script>`
- Test it
  - **`gulp`**
  - **`node server.js`**

  these need to be run from separate command prompts
  - browse **`localhost:3000`**
  - modify **`index.html`** and verify that browser is updated
  - modify **`notes.css`** and verify that csslint is run and browser is updated
  - modify **`notes.js`** and verify that jshint is run and browser is updated