ADALAB

creando diversidad digital

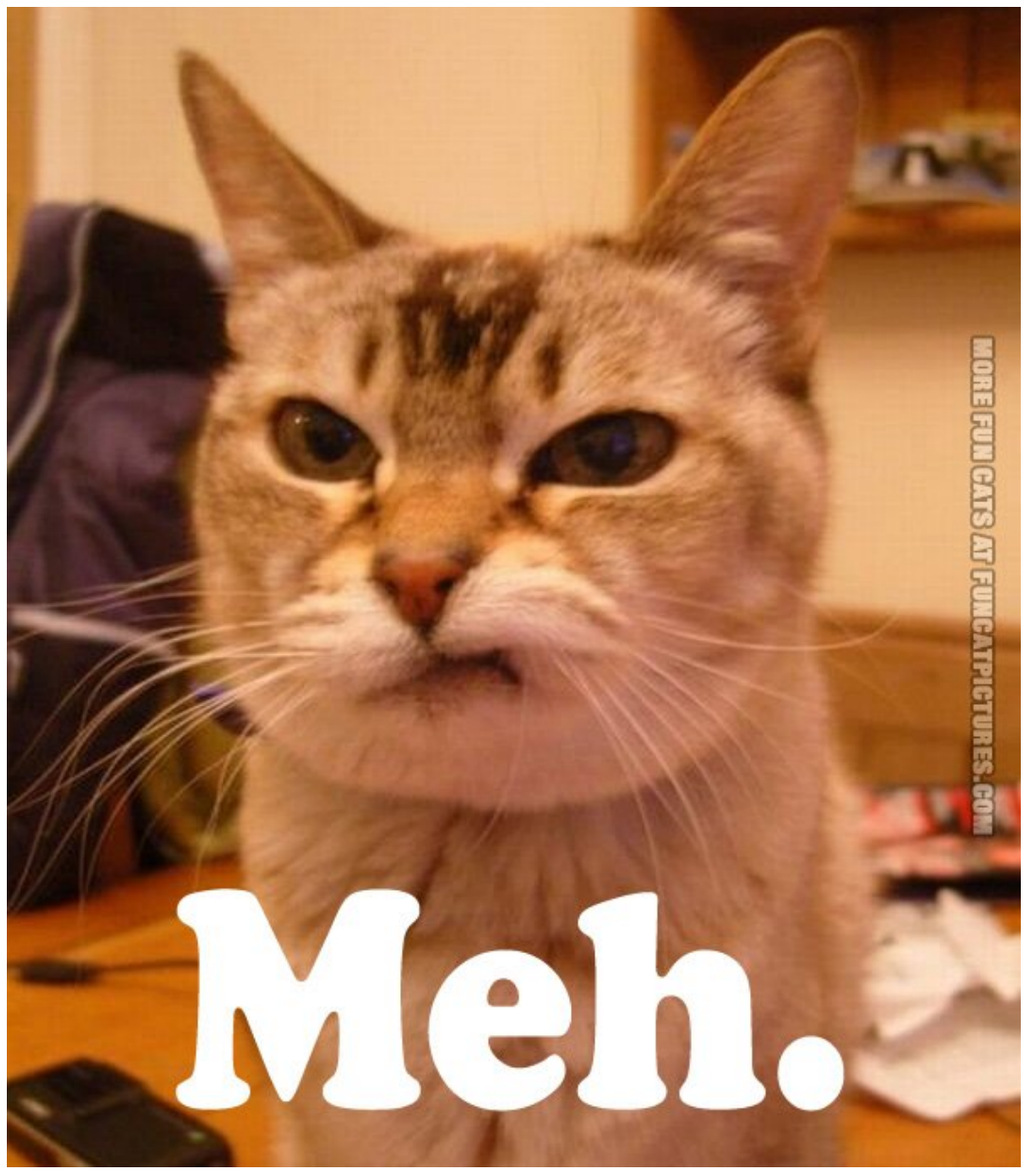
INTRO A GULP

AUTOMATIZACIÓN DE TAREAS EN PROYECTOS DE FRONTEND

Luis de Dios
@luisddm_

EL PROBLEMA

- Cuando desarrollamos software tenemos que hacer tareas
  - **tediosas y repetitivas**
  - siempre se **olvidará** alguna en algún momento si hay que hacerlas a mano
  - **desmotivantes**



¿POR QUÉ HAY QUE HACER ESTE TIPO DE TAREAS?

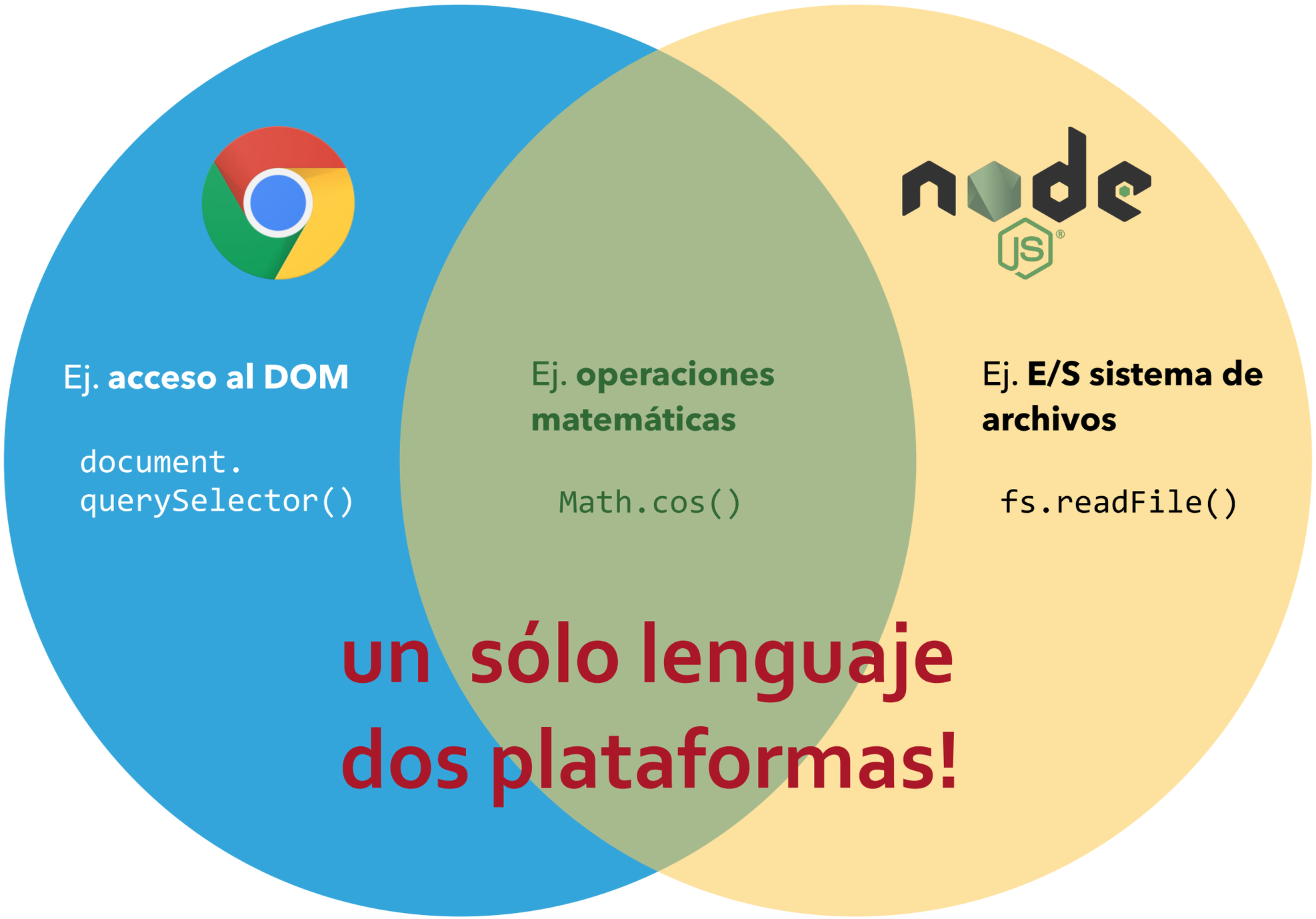
- Queremos conseguir dos cosas al mismo tiempo
  - que nuestro código sea **limpio, legible y estructurado** cuando lo leemos y escribimos
  - pero también que sea **rápido y eficiente** en la descarga y ejecución
- Esto implica **transformaciones** continuas

¿QUÉ ES NODE?

- Es una **plataforma JavaScript** construida sobre V8
- Nos permite **ejecutar código JavaScript fuera del navegador**
- Node viene con **npm, un gestor de paquetes** que permite usar pequeños componentes modulares para construir grandes aplicaciones

plataforma ≠ lenguaje

NAVEGADOR VS NODE

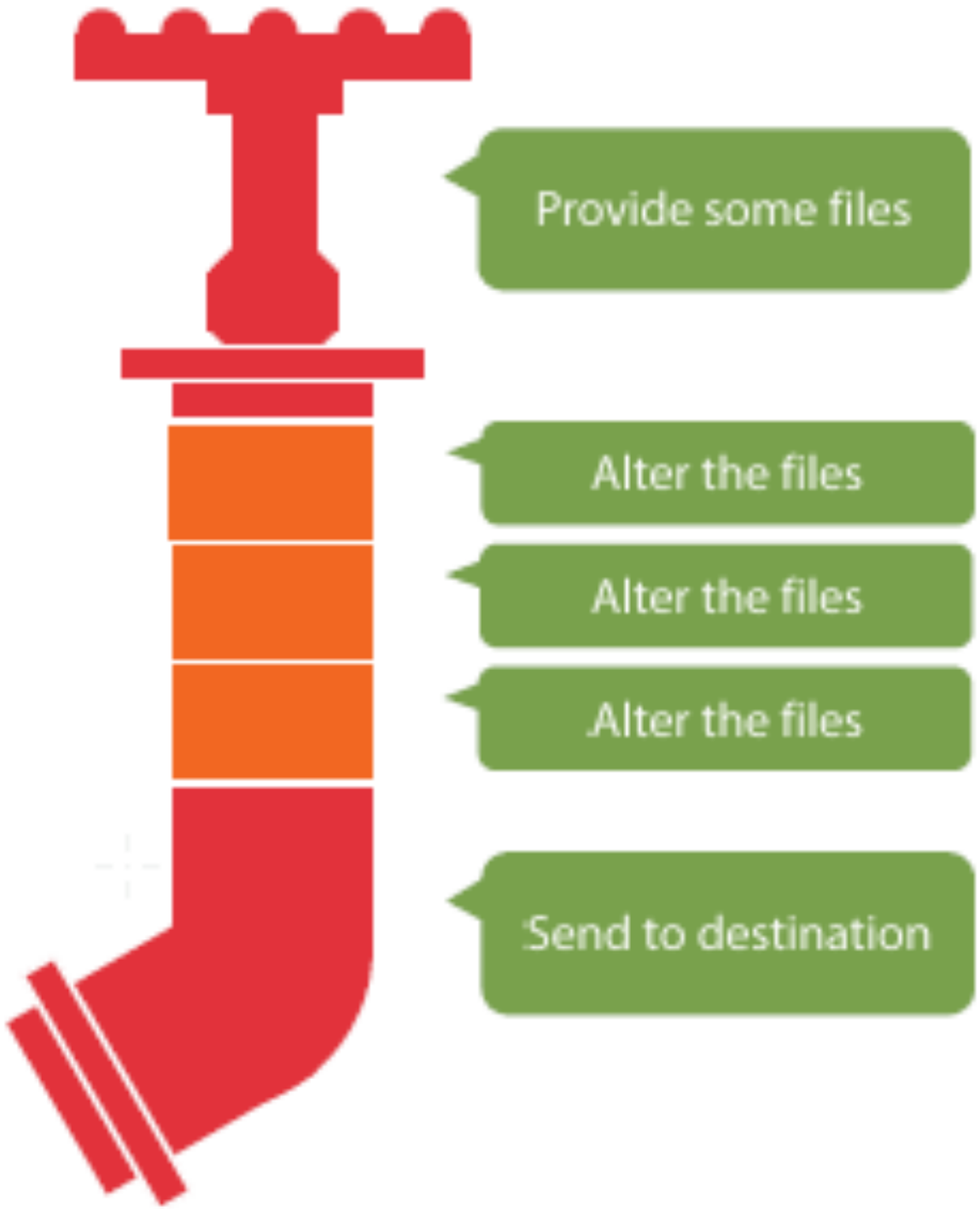


¿QUÉ ES GULP?

- Gulp es un **automatizador de tareas** hecho en **JavaScript**.
- Utiliza el concepto de **streams** para ejecutar tareas rápida y eficientemente.
- Sólo requiere **un pequeño fichero con instrucciones** para llevar a cabo cualquier tarea.
- Gulp usa **plugins pequeños y con un único propósito** para procesar ficheros. Se suelen **encadenar** para hacer cosas más complejas.
- Gulp se ejecuta sobre **Node** y utiliza **npm** para gestionar plugins.

EL FLUJO DE DATOS EN GULP

- Usamos **plugins** de Gulp
- Cada uno de ellos hace una tarea **concreta** y **sencilla**
- Pero podemos **encadenarlos** para hacer cosas complejas
- A partir de una entrada, se aplica una **transformación** y se devuelve una salida



PLUGINS DE GULP COMO DEPENDENCIAS DE NODE

- Necesitamos mantener un listado de las **dependencias** que usamos en nuestro proyecto para poder replicarlo en cualquier máquina.
- Estas dependencias se instalan con **npm**.
- Pueden ser plugins de Gulp, pero también cualquier otra cosa.

http://gulpjs.com/plugins

package.json

```
"dependencies": {
  "gulp": "^3.9.1",
  "gulp-babel": "^6.1.2",
  "gulp-clean-css": "^2.0.11",
  "gulp-concat": "^2.6.0",
  "gulp-gzip": "^1.4.0",
  "gulp-html-replace": "^1.6.1",
  "gulp-sass": "^2.3.2",
  "gulp-tar": "^1.9.0",
  "gulp-uglify": "^1.5.4"
}
```

ARCHIVO DE CONFIGURACIÓN DE GULP

- Aparte del `package.json`, necesitaremos un **`gulpfile.js`** que será lo que Gulp ejecute.
- Contiene todas y cada una de la tareas que podremos ejecutar.
- El **`gulpfile.js`** estará escrito en JavaScript y será ejecutado sobre Node en nuestra máquina.

PLANTILLA DE UNA TAREA

```
gulp.task('task', function () {
  gulp.src('source')
    .pipe(plugin1())
    .pipe(plugin2())
    .pipe(gulp.dest('destination'))
});
```

ESCRIBIENDO UNA TAREA

```
gulp.task('task', function () {
  gulp.src('source')
    .pipe(plugin1())
    .pipe(plugin2())
    .pipe(gulp.dest('destination'))
});
```

```
gulp.task(string, function)
```

- **Nombra y determina cada tarea**
  - el string es el nombre de la tarea
  - la función es un *callback* que contiene el código a ejecutar

ESCRIBIENDO UNA TAREA

```
gulp.task('task', function () {
  gulp.src('source')
    .pipe(plugin1())
    .pipe(plugin2())
    .pipe(gulp.dest('destination'))
});
```

```
gulp.src(string || array<string>)
```

- **Indica el archivo o los archivos de entrada a procesar**

ESCRIBIENDO UNA TAREA

```
gulp.task('task', function () {
  gulp.src('source')
    .pipe(plugin1())
    .pipe(plugin2())
    .pipe(gulp.dest('destination'))
});
```

```
function.pipe(function)
```

- **Ejecuta transformaciones sobre los ficheros de entrada y devuelve un fichero de salida**
  - la función viene de un plugin de Gulp
- Cada pipe sería un segmento de la tubería y permite el encadenamiento de funciones

ESCRIBIENDO UNA TAREA

```
gulp.task('task', function () {
  gulp.src('source')
    .pipe(plugin1())
    .pipe(plugin2())
    .pipe(gulp.dest('destination'))
});
```

```
gulp.dest(string)
```

- **Indica a qué fichero irán a parar los datos una vez procesados**

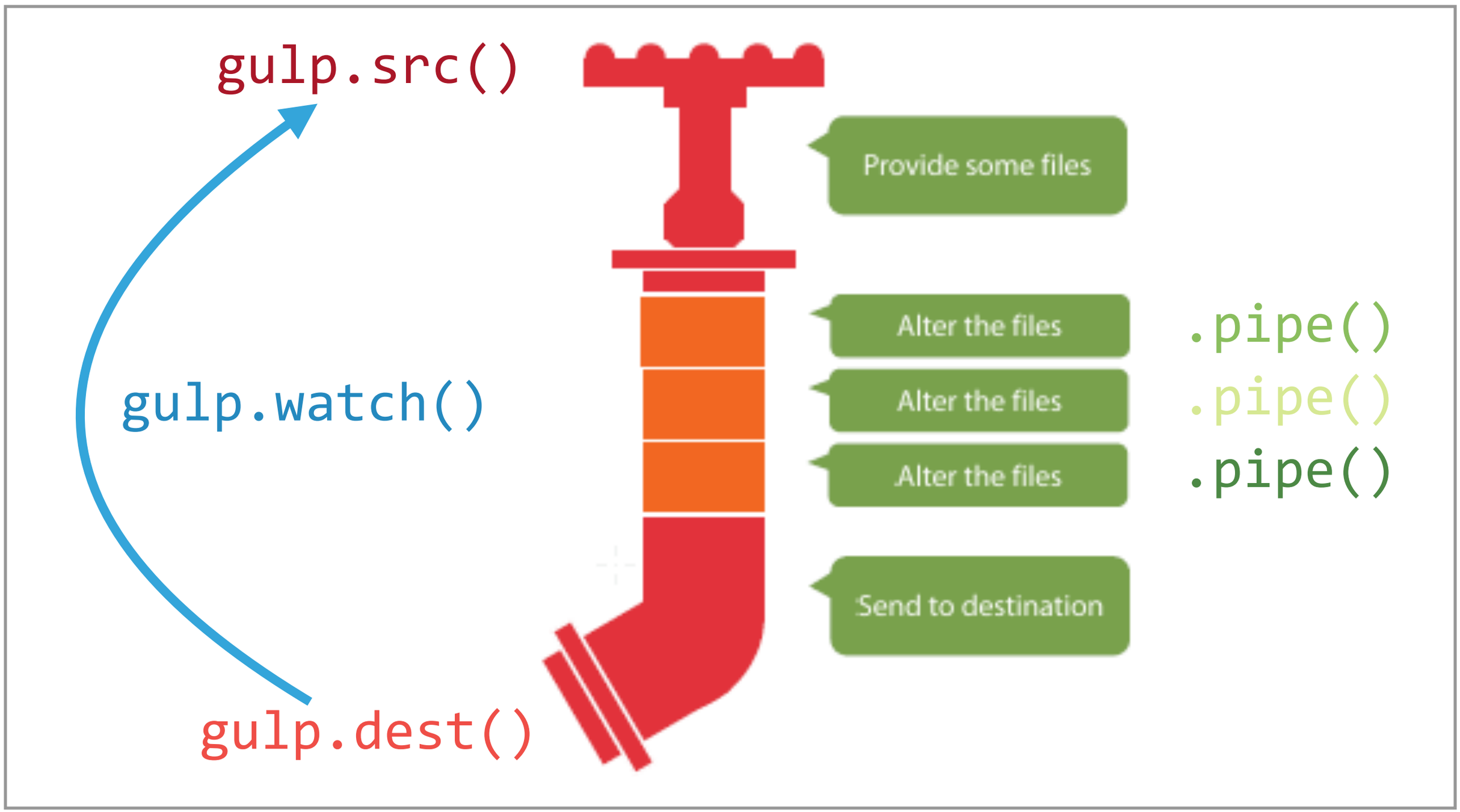
ESCRIBIENDO UNA TAREA

```
gulp.watch('source', ['task']);
```

```
gulp.watch(string, array<string>)
```

- **Busca cambios en los archivos de entrada**
  - el string es la ruta del fichero o los ficheros a observar
  - el array son las tareas a ejecutar cuando se detecta un cambio en alguno de ellos

EL CICLO DE GULP



```
const sass        = require('gulp-sass');
const uglify      = require('gulp-uglify');
const cleanCss    = require('gulp-clean-css');
const htmlReplace = require('gulp-html-replace');
const babel       = require('gulp-babel');
const tar         = require('gulp-tar');        // https://www.npmjs.com/package/gulp-tar
const gzip        = require('gulp-gzip');       // https://www.npmjs.com/package/gulp-gzip

gulp.task('scss', () =>
  gulp.src('src/scss/styles.scss')
    .pipe(sass().on('error', sass.logError))
    .pipe(cleanCss())
    .pipe(gulp.dest('dist/css'))
);

gulp.task('es6', () =>
  gulp.src('src/js/*.js')
    .pipe(concat('scripts.js'))
    .pipe(babel({ presets: ['es2015'] }))
    .pipe(uglify())
    .pipe(gulp.dest('dist/js'))
);

gulp.task('html', () =>
  gulp.src('src/index.html')
    .pipe(htmlReplace({
      css: 'css/styles.css',
      js: 'js/scripts.js',
    }))
    .pipe(gulp.dest('dist'))
);

gulp.task('compress', () =>
  gulp.src('dist/*')
    .pipe(tar('code.tar'))   // Pack all the files together
    .pipe(gzip())            // Compress the package using gzip
    .pipe(gulp.dest('.'))
);
```

LET'S CODE!

¿Y PARA QUÉ PUEDE SERVIR GULP?

- transpilar, concatenar, minificar, optimizar imágenes, copiar, mover, eliminar y renombrar archivos, compilar SASS, autoprefijar CSS, generar source maps, ejecutar tests, lintear, generar informes, gestionar git, empaquetar y comprimir, subir a producción, recarga automática del navegador cuando se producen cambios, etc.

TALLER

- **Vamos a hacer algunas de estas tareas como ejemplo.**

AUTORELOAD

Observa `/src`

Cambios en el código **fuente**

Procesa automáticamente la entrada y genera una salida

Abre un **servidor web local**

Observa `/dist`

Cambios en el código **generado**

Recarga automáticamente el navegador

CONCLUSIONES

- **Compromiso entre rapidez y flexibilidad**. Gulp aporta flexibilidad a expensas de necesitar una configuración detallada.

- Hay **otras formas** de hacer lo que hace Gulp
  - usando **Grunt, Broccoli, Webpack**, etc., que son similares aunque tienen diferentes orientaciones
  - con los **CLIs** (ej. `create-react-app`), que se basan en los anteriores y funcionan *out-of-the-box*

- **¡No hay magia!**

github.com/luisddm

gulp-adalab

@luisddm_