Sesión 1

PostCSS

Entornos de desarrollo

escuelaIT

Sesiones del curso

Sesiones del curso

- **SESIÓN 0 - PostCSS: Lo que necesitas saber** (Sesión en abierto)
  Origen, ¿Qué es PostCSS?, ¿Que no es PostCSS?

- **SESIÓN 1 - Entornos de desarrollo**
  Herramientas para usar PostCSS, Codepen, Prepros, Gulp, Grunt, npm

- **SESIÓN 2 - De Sass a PostCSS**
  Cómo configurar PostCSS para desarrollar como si fuera Sass

- **SESIÓN 3 - CSS del futuro**
  Cómo configurar PostCSS para desarrollar con la próxima generación de CSS

- **SESIÓN 4 - Plugins**
  Los plugins más conocidos e interesantes y aprenderemos a crear nuestro propio plugin PostCSS

Agenda

Abril

18	19	20	21	22
SESIÓN 0 - PostCSS: Lo que necesitas saber Sesión en abierto		**SESIÓN 1 - Entornos de desarrollo**	**SESIÓN 2 - De Sass a PostCSS**	**SESIÓN 3 - CSS del futuro**
25	26	27	28	29
SESIÓN 4 - Plugins				

escuelaIT

¿Qué es **PostCSS**?

¿**Qué** es?

"PostCSS es una **herramienta para transformar CSS** con plugins de JavaScript".

La herramienta de por sí no lo transforma, sino que lo convierte en un formulario de datos que JavaScript puede manipularlo mediante plugins.

¿Qué hacen los plugins?

Los plugins de PostCSS pueden:

- Comportarse como preprocesadores
- Optimizar y añadir prefijos al código
- Añadir la futura sintaxis
- Supervisar el código
- atajos de código...

la lista es larga y variable.

Sin **límites**

No hay límites en el tipo de manipulación que los plugins de PostCSS pueden aplicar al CSS.

Si puedes pensarlo, probablemente puedes escribir un plugin de PostCSS para hacer que suceda.

Demo

Ventajas

Ventajas

- Versatilidad
- Modularidad y flexibilidad
- Rapidez
- DIY
- CSS regular
- Se utiliza con entornos de desarrollo comunes

Entornos de desarrollo

Varios **entornos** para elegir

CODEPEN

Prepros

gulp

GRUNT

npm

CODEPEN

Pen Settings

HTML **CSS** JavaScript Behavior

CSS Preprocessor Need an add-on?

PostCSS

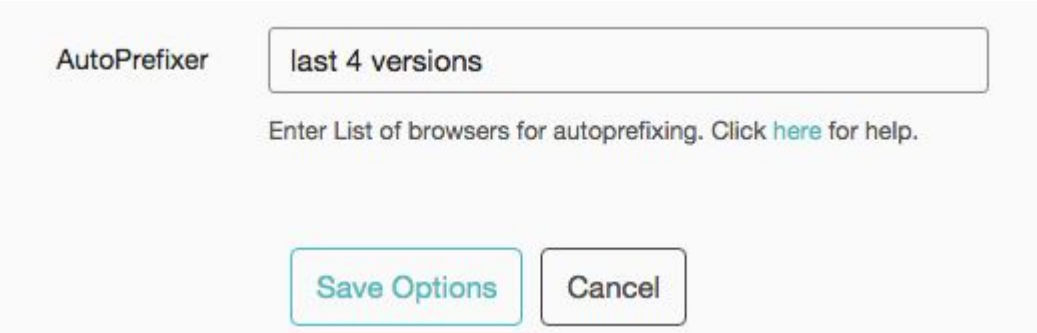
PLUGINS FOR POSTCSS

Plugins for PostCSS are handled through the postcss-use plugin, so that you can call them directly from your code.

Search add-ons

@use cssnext;	Add
@use postcss-simple-vars;	Add
@use postcss-discard-comments;	Add
@use postcss-custom-media;	Add

Prepros


AutoPrefixer
last 4 versions
Enter List of browsers for autoprefixing. Click here for help.
Save Options
Cancel

Pero si queremos **controlar los plugins** que queremos usar necesitamos un task runner

Task **runners**

gulp

GRUNT

La instalación

Requisitos previos

Instalar Node

- Instalar desde la página oficial Nodejs.org
- Test: Run `node -v` La versión debería ser mayor que v0.10.32.
- Si ya estamos usando una versión de Node que queremos mantener, podemos usar gestores de versiones como `n` (https://github.com/tj/n) que nos permitirá saltar de una versión a otra.

Actualizar npm

Aunque node ya viene con el **npm** (Node Package Manager) instalado, éste último se actualiza más a menudo y es **recomendable actualizarlo**.

- **sudo npm install npm -g**
- Test: Run **npm -v.** La versión debería ser mayor que 2.1.8.

Gulp

gulp

Instalar Gulp

- Instalar Gulp globalmente con:
  - [sudo] npm install gulp -g

Configuración de un Proyecto para Gulp

- Agregar el archivo package.json
  - npm init

- Instalar Gulp Package (Paquete Gulp)
  - npm install gulp --save-dev

- Agregar gulpfile.js
  - Al directorio raíz de tu Proyecto agrega un archivo llamado "gulpfile.js"

Configuración básica de Gulp PostCSS

Dentro de la carpeta del proyecto crear **dos subcarpetas**:

- "src"
- "dest"

La carpeta "src" tendrá los archivos CSS sin procesar, mientras que la carpeta "dest" tendrá los archivos PostCSS procesados.

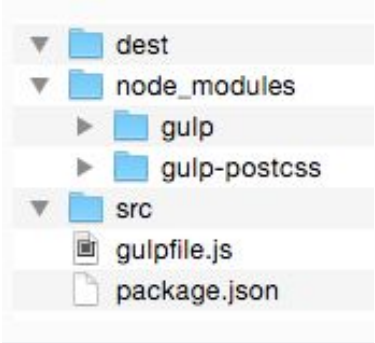
Configuración básica de Gulp PostCSS

Instalar el plugin de gulp-postcss en el proyecto

```
npm install --save-dev gulp-postcss
```

Configuración básica de Gulp PostCSS

Después de que la instalación se complete, la estructura del proyecto debería tener este aspecto:



Editar el gulpfile

Abrir el **gulpfile.js** crear variables para llamar los módulos "gulp" y "gulp-postcss" añadiendo el siguiente código:

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

Editar el gulpfile

Ahora podemos configurar una tarea para leer un archivo fuente CSS y procesarlo con PostCSS. Añade lo siguiente:

```
gulp.task('css', function () {
  var processors  = [
  ];
  return gulp.src('./src/*.css')
    .pipe(postcss(processors))
    .pipe(gulp.dest('./dest'));
});
```

Test

Crear un nuevo archivo "**style.css**" en la carpeta "**src**" y añadir un poco de CSS prueba como este:

```
.container{
        display:flex;
}
```

En la terminal, ubicada en la carpeta del proyecto, ejecutar el comando:

```
gulp css
```

Añadir plugins PostCSS

Ahora vamos a añadir una selección de **plugins** PostCSS y paquetes:

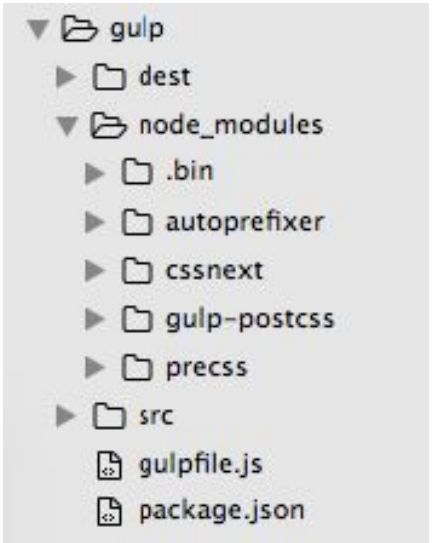
- Autoprefixer (agrega prefijos del proveedor),
- cssnext (habilita sintaxis futura)
- precss (extiende la funcionalidad como Sass)

Añadir plugins PostCSS

```
npm install autoprefixer --save-dev
npm install cssnext --save-dev
npm install precss --save-dev
```

Añadir plugins PostCSS

Después de la instalación de plugins la arquitectura de carpetas queda:



Editar el gulpfile

```
var autoprefixer = require('autoprefixer');
var cssnext = require('cssnext');
var precss = require('precss');
```

Añadir estos tres plugins al array **processors** en nuestra tarea de gulp.

```
var processors = [
  autoprefixer,
  cssnext,
  precss
];
```

Testing

```
/* Testing autoprefixer */
.autoprefixer {
  display: flex;
}
/* Testing cssnext */
.cssnext {
  background: color(red alpha(-10%));
}
/* Testing precss */
.precss {
  @if 3 < 5 {
    background: green;
  }
  @else {
    background: blue;
  }
}
```

```
/* Testing autoprefixer */
.autoprefixer {
  display: -webkit-box;
  display: -webkit-flex;
  display: -ms-flexbox;
  display: flex;
}

/* Testing cssnext */

.cssnext {
  background: rgba(255, 0, 0, 0.9);
}

/* Testing precss */

.precss {
  background: green
}
```

Configuración del plugin

```
var processors = [
  autoprefixer({browsers: ['last 3 version']}),
  cssnext,
  precss
];
```

Watch

```
// Watch
gulp.task('watch', function() {
    gulp.watch('src/**/*.css', ['css']);
});

// Default
gulp.task('default', ['css', 'watch']);
```

Grunt

Instalar Grunt

- Instalar Grunt globalmente con:
  - [sudo] npm install -g grunt-cli

Configuración de un Proyecto para Grunt

- Agregar el archivo package.json
  - npm init

- Instalar Grunt Package
  - npm install grunt --save-dev

- Agregar gruntfile.js
  - Al directorio raíz de tu Proyecto agrega un archivo llamado "gruntfile.js"

Configuración básica de Grunt PostCSS

Dentro de la carpeta del proyecto crear **dos subcarpetas**:

- "src"
- "dest"

La carpeta "src" tendrá los archivos CSS sin procesar, mientras que la carpeta "dest" tendrá los archivos PostCSS procesados.

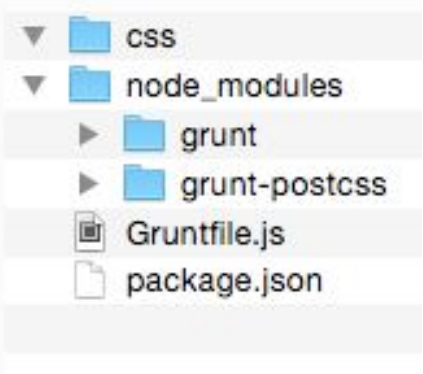
Configuración básica de Grunt PostCSS

Instalar el plugin de grunt-postcss en el proyecto

```
npm install --save-dev grunt-postcss
```

Configuración básica de Grunt PostCSS

Después de que la instalación se complete, la estructura del proyecto debería tener este aspecto:



Editar el gruntfile

En el **gruntfile.js** añadir la estructura básica que todos los proyectos gulp necesitan:

```
module.exports = function(grunt) {


};
```

Después, añadir la task dentro de la estructura creada:

```
grunt.loadNpmTasks('grunt-postcss');
```

Editar el gruntfile

Añadir grunt.initConfig justo encima donde hemos añadido grunt.loadNpmTasks, nos queda:

```
module.exports = function(grunt) {
    grunt.initConfig({

 });
    grunt.loadNpmTasks('grunt-postcss');
};
```

Editar el gruntfile

Dentro de él, configuramos un objeto llamado postcss así:

```
module.exports = function(grunt) {
  grunt.initConfig({
    postcss: {

    }
  });
  grunt.loadNpmTasks('grunt-postcss');
};
```

Editar el gruntfile

Dentro del nuevo objeto `postcss` agregaremos dos objetos anidados, uno llamado `options` y otro `dist`.

```
module.exports = function(grunt) {
 grunt.initConfig({
   postcss: {
    options: {
    },
    dist: {
    }
   }
   });
 grunt.loadNpmTasks('grunt-postcss');
};
```

Editar el gruntfile

Dentro del objeto `options`, agregue un array vacío llamado `processors`. Este es donde configuramos los plugins PostCSS para usarlos un poco más adelante. Por ahora, sólo lo actualizamos a:

```
options: {
 processors: [
 ]
},
```

Editar el gruntfile

Ahora actualiza el objeto `dist` para especificar "src/style.css" como nuestro archivo fuente, y "dest/style.css" como el archivo que queremos generar:

```
dist: {
  src: 'src/style.css',
  dest: 'dest/style.css'
}
```

Añadir style.css dentro de la carpeta src del proyecto.

Test

Hacer testeo rápido para ver que en la carpeta dest se replica el style.css

grunt postcss

Añadir plugins PostCSS

Ahora vamos a añadir una selección de **plugins** PostCSS y paquetes:

- Autoprefixer (agrega prefijos del proveedor),
- cssnext (habilita sintaxis futura)
- precss (extiende la funcionalidad como Sass)

Añadir plugins PostCSS

```
npm install autoprefixer --save-dev
npm install cssnext --save-dev
npm install precss --save-dev
```

Editar el gulpfile

Añadir estos tres plugins al array **processors** en nuestra tarea de grunt.

```
processors: [
  require('autoprefixer')(),
  require('cssnext')(),
  require('precss')()
]
```

Testing

```
/* Testing autoprefixer */
.autoprefixer {
  display: flex;
}
/* Testing cssnext */
.cssnext {
  background: color(red alpha(-10%));
}
/* Testing precss */
.precss {
  @if 3 < 5 {
    background: green;
  }
  @else {
    background: blue;
  }
}
```

```
/* Testing autoprefixer */
.autoprefixer {
  display: -webkit-box;
  display: -webkit-flex;
  display: -ms-flexbox;
  display: flex;
}

/* Testing cssnext */

.cssnext {
  background: rgba(255, 0, 0, 0.9);
}

/* Testing precss */

.precss {
  background: green
}
```

Configuración del plugin

```
processors: [
 require('autoprefixer')({browsers: ['last 1 version']}),
 require('cssnext')(),
 require('precss')()
]
```

npm

Lanzar postCSS mediante NPM

- Agregar el archivo package.json
  - npm init

También podemos lanzar postCSS mediante un script de NPM.
Para ello vamos a instalar postcss-cli.

```
npm install postcss-cli --save-dev
```

Añadir plugins PostCSS

```
npm install autoprefixer --save-dev
npm install cssnext --save-dev
npm install precss --save-dev
```

Archivo de configuración

Con este método también hay que crear un archivo de configuración.

- Para eso creamos un nuevo archivo en la raiz llamada **config-postcss.json**

Archivo de configuración: config-postcss.js

```
{
  "use": [
    "autoprefixer",
    "cssnext",
    "precss"
  ],
  "autoprefixer": {
      "browsers": ["last 1 version"]
  },
  "input": "src/style.css",
  "output": "dest/style.css"
}
```

Archivo de configuración: package.json

En el package.json vamos a crear un script con el nombre **build:css** en la sección de scripts.

```
"scripts": {
    "build": "postcss -c config-postcss.json --no-map.inline",
    "watch": "postcss -c config-postcss.json -w --no-map.inline"
    }
```

Lanzar el conjuro

npm run build
npm run watch