**¿Qué es PostCSS ?**

**PostCSS:**Es un post-procesador de CSS, que a diferencia de los pre-procesadores, se apoya de JavaScript para tener más control sobre el archivo CSS resultante.

Cuenta con una gran comunidad y cantidad a de plugins, entre ellos, CSS-Next, el cuál, nos da acceso a la futura versión de CSS compatible ya mismo en nuestros proyectos.

Entonces, un post-procesador a diferencia de un pre-procesador, no es un cambio de sintaxis, es una nueva plataforma para crear estilos como nunca antes.

**Nomenclatura BEM ( Bloque, Elemento, Modificador )**

Antes de iniciar a usar PostCSS en nuestro proyecto, vamos a modificar el código de Invie a una nomenclatura de clases llamada BEM.

BEM significa Bloque, Elemento, Modificador, y nos permite separar por bloques, cada parte del código, haciéndolo mucho más legible.

**Configurando nuestras tareas en Gulp: Servidor, Procesamiento de CSS y Watch**

Gulp será el gestor de tareas que nos ayudará a orquestar todo lo que hagamos con PostCSS.

La primera, será una tarea que sirve como servidor de desarrollo:

gulp.task(‘serve’,**function** () {

browserSync-**init**({

**server**: {

baseDir: ‘./dist’

}

})

})

La segunda, será la tarea para procesar el CSS:

gulp.task(‘css’,**function** () {

**var** processors = []

**return** gulp.src(‘./src/\*.css’)

.pipe(**postcss**(**processor**))

.pipe(postcss(processor))/css’’))

.pipe(**browserSync**.stream())

})

La última, será la tarea para observar los cambios:

gulp.task(‘watch’,**function** () {

gulp.watch(‘./src/\*css’, [‘css])

gulp.watch(‘./dist/\*.html’).**on**(‘change’, browserSync.reload)

})

Por último, para unificar las 3 tareas usamos:

**gulp**.task(‘**default**’, [‘watch’, ‘serve’])