POSTCSS

**Todo un set de herramientas y plugins para transformaciones del CSS mediante Javascript útil para proyectos frontend.**

PostCSS es una herramienta moderna para la gestión del código CSS que permite aumentar la productividad a la vez que libera al desarrollador de trabajos adicionales así como conocimientos técnicos del estado actual del lenguaje y el soporte en los navegadores.

PostCSS por si solo no hace trabajo alguno sobre el CSS. En cambio es necesario instalar plugins que nos permitan indicar qué tipo de transformaciones serán realizadas sobre el CSS que tenemos de entrada, generando un CSS de salida acorde con las necesidades de cada proyecto o las tecnologías con las que se trabaje.

Existen más de 200 plugins de PostCSS para realizar todo tipo de transformaciones, que se pueden encontrar en la [página de GitHub de PostCSS](https://github.com/postcss/postcss).

PostCSS se puede integrar con herramientas frontend existentes en un proyecto dado, como pueden ser Webpack o Parcel, o también se puede usar directamente por medio de su propia CLI.

**Menu**

* [**Instalar PostCSS**](https://desarrolloweb.com/home/postcss#track163)
* [**Configuración de PostCSS**](https://desarrolloweb.com/home/postcss#track164)
* [**Uso de PostCSS**](https://desarrolloweb.com/home/postcss#track165)
* [**Lanzar el proceso de build de manera automática**](https://desarrolloweb.com/home/postcss#track166)
* **[Plugin de importado de PostCSS](https://desarrolloweb.com/home/postcss" \l "track167)**
* **[Plugin PurgeCSS](https://desarrolloweb.com/home/postcss" \l "track219)**
* **[Plugin de PostCSS para minimizar el CSS](https://desarrolloweb.com/home/postcss" \l "track236)**
* **[Plugin para anidación de reglas CSS](https://desarrolloweb.com/home/postcss" \l "track339)**

## **Instalar PostCSS**

PostCSS se instala en el proyecto donde lo desees utilizar. Es una herramienta que funciona bajo [NodeJS](https://desarrolloweb.com/home/nodejs), por lo que previamente tendrás que haber instalado esa plataforma en tu equipo. Por cierto, PostCSS también es compabible con [Deno](https://desarrolloweb.com/home/deno).

Lo puedes instalar como un añadido a cualquier sistema de build que vengas utilizando, como podría ser [Webpack](https://desarrolloweb.com/manuales/manual-webpack.html), aununque también lo puedes usar de manera autónoma gracias a su CLI. Ahora vamos a explicar cómo instalar PostCSS con PostCSS-CLI.

Lo instalas vía npm, por lo que necesitas haber inicializado el proyecto con npm:

npm init

Vas contestando el asistente de inicialización y luego tienes que instalar dos dependencias, como dependencias de desarrollo, en el proyecto:

npm install -D postcss postcss-cli

Además, necesitas instalar todos los plugins que consideres necesarios de PostCSS para tu proyecto. Una alternativa muy popular es "autoprefixer", que se encarga de procesar tu CSS y colocar los prefijos en los atributos CSS que sean necesarios.

npm install -D autoprefixer

Este paso de instalación lo debes completar luego con la configuración de PostCSS, en la que indicarás qué plugins se deben usar.

## **Configuración de PostCSS**

Una vez tienes instaladas todas las dependencias, incluido el número de plugins que veas conveniente, debes crear un archivo de configuración de PostCSS, llamado "postcss.config.js". Ese archivo contendrá el código Javascript que NodeJS debe utilizar para configurar la ejecución de PostCSS. En él básicamente indicarás los plugins que se van a usar y su configuración.

Un ejemplo sencillo de configuración de PostCSS en postcss.config.js sería el siguiente:

module.exports = {

plugins: [

require('autoprefixer'),

]

}

Como puedes apreciar, PostCSS se ejecuta con NodeJS, por eso usamos la sentencia "module.exports", que es la manera de exponer un módulo de Javascript hacia afuera que se usa en NodeJS, que se llama "CommonJS**"**

## **Uso de PostCSS**

Para comenzar a usar PostCSS nos queda un último paso, que es crear un script de npm para ejecutar PostCSS cada vez que necesites procesar tu CSS. Ese script de npm lo tienes que colocar dentro del archivo "package.json".

El archivo "package.json", tiene una propiedad llamada "scripts", en la que puedes colocar cualquier número de scripts npm que puedas necesitar en un proyecto. Su forma sería como esta:

"scripts": {

"build": "postcss src/css/estilos.css --output dist/estilos.css"

},

En nuestro caso hemos creado un script llamado "build", que hace la invocación al CLI de PostCSS, indicando dos informaciones:

* **src/css/estilos.css sería la ruta inicial donde se encuentra el archivo CSS que se desea procesar.**
* **--output dist/estilos.css sería la ruta de destino donde se colocará el CSS una vez procesado.**

Una vez creado el script npm, lo ejecutas con la consola, desde la raíz de tu proyecto con el comando "npm run" seguido del nombre del script que hayas puesto.

npm run build

Esto realizará el procesamiento del archivo de origen, aplicando todas las transformaciones de todos los plugins que hayas configurado, y escribiendo el archivo de destino en la ruta indicada.

**Por ejemplo, si tuvieras un CSS como este:**

myGrid {

display: grid;

grid-template-columns: 12px 12px 12px;

grid-template-rows: 12px 12px 12px;

column-gap: 1rem;

}

**Lo que tendríamos como salida es esto:**

**.** myGrid {

display: grid;

grid-template-columns: 12px 12px 12px;

grid-template-rows: 12px 12px 12px;

-moz-column-gap: 1rem;

column-gap: 1rem;

}

Sin embargo ten en cuenta que esta salida puede ser variable, ya que autoprefixer tiene en cuenta un rango de navegadores de destino y, a medida que los navegadores se actualizan hay ciertos prefijos que ya no son necesarios.

Si deseas cambiar los navegadores de destino puedes hacerlo de diversas maneras, por ejemplo creando un archivo ". browserslistrc" en la raíz de tu proyecto. Mira la documentación para encontrar más detalles.

## **Lanzar el proceso de build de manera automática**

En tiempo de desarrollo realizamos cambios en el código CSS con mucha frecuencia. Como nuestro código CSS es generado, para ver los cambios en el CSS necesitaríamos lanzar el proceso de build, con el comando que ejecuta el correspondiente script npm.

Resulta muy incómodo tener que lanzar el proceso de build con cada cambio que hagamos para poder ver el nuevo aspecto que tiene nuestra página en el navegador, así que lo ideal es crear un sistema de "watch" que esté pendiende de los cambios en los archivos de origen, para compilarlos de nuevo automáticamente cada vez que se modifican.

Esto se puede configurar de una manera muy sencilla, con el flag --watch en la llamada a PostCSS. Lo ideal es tener un script npm para el build y otro para el watch, lo que nos quedaría más o menos así.

"scripts": {

"build": "postcss src/css/styles.css --output dist/css/styles.css",

"watch": "postcss src/css/styles.css --output dist/css/styles.css --watch"

},

Ahora, para lanzar el proceso de watch y realizar cambios automáticamente en el CSS generado ejecutaremos el comando siguiente:

npm run watch

## **Plugin de importado de PostCSS**

Una de las utilidades más interesantes de PostCSS es la capacidad de hacer imports de distintos archivos de CSS dentro del CSS principal del proyecto. De este modo podemos desarrollar el código CSS de un proyecto en varios archivos pequeños, en lugar de una grande y difícil de mantener.

Para hacer el importado de otros archivos CSS se usa el plugin postcss-import. Se debe de instalar así:

npm install -D postcss-import

Una vez instalado lo podemos incluir en nuestro archivo de configuración postcss.config.js.

module.exports = {

plugins: [

require('postcss-import'),

// otros plugins...

]

}

Generalmente lo querrás incluir como primer elemento, para construir un gran archivo CSS con todo el código junto, antes de pasarle las siguientes transformaciones.

Los imports en CSS se realizan con la directiva @import, indicando la ruta donde esta el archivo que deseas importar. Preprocesadores como Sass usan la misma técnica. El código CSS sería más o menos como este:

@import "./componentes/navegador.css";

A la hora de importar otros archivos de código CSS necesitas tener en cuenta:

* Los imports se deben realizar siempre al principio de los archivos
* Para organizar tus imports es muy habitual hacer un archivo raíz en el que no tengas CSS, solo los imports de otros archivos.
* Puedes importar cosas instaladas en node-modules, simplemente usando los nombres de los packages y la ruta del archivo dentro de ese package.

## **Plugin PurgeCSS**

El plugin PurgeCSS permite eliminar de un archivo de código CSS todas las reglas de estilo que no se están utilizando. Al ponerse en marcha este plugin de PostCSS se hace un análisis del código CSS y posteriormente el código de la página o aplicación web, realizando transformaciones en el código CSS para eliminar todas las reglas de selectores que no se están usando en el código.

Es un plugin esencial cuando se trabaja con frameworks CSS como Bootstrap o [Tailwind](https://desarrolloweb.com/home/tailwind-css), puesto que estos frameworks incluyen mucho CSS que en realidad no se llega a usar en los proyectos. Gracias a este plugin podemos asegurar que el CSS que se lleva a producción incluye única y exclusivamente código CSS que se usa en el sitio web.

Se instala de la siguiente manera:

npm i -D @fullhuman/postcss-purgecss

Luego tienes que configurar el plugin dentro del archivo postcss.config.js, indicando con una expresión regular los archivos donde se encuentra el código HTML de tu proyecto, es decir todos los ficheros donde puede haber código HTML, para que lo analice en función del código CSS del proyecto, para saber todos los selectores que verdaderamente importan para el sitio web.

En un sitio web sencillo podríamos tener solamente páginas HTML en una carpeta, pero podrías tener templates en archivos Javascript o código HTML en archivos .php, por ejemplo. Para indicar todos los tipos de archivos y carpetas donde buscar se indican distintos patrones de rutas en el array "content".

El código para usar PurgeCSS nos quedaría así:

const purgecss = require('@fullhuman/postcss-purgecss');

module.exports = {

plugins: [

purgecss({

content: [

'./public/\*\*/\*.html',

'./public/\*\*/\*.php',

'./public/js/\*\*/\*.js',

]}),

]

}

Una vez modificado el archivo de configuración de PostCSS lanzamos el proceso de ejecución de PostCSS de manera habitual y observaremos que el código CSS generado habrá sido reducido a lo esencial. Por supuesto, dependiendo de si usamos frameworks o no puede que la diferencia de peso sea más o menos notoria.

## **Plugin de PostCSS para minimizar el CSS**

Gracias a PostCSS podemos conseguir realizar cómodamente otra de las tareas que nos ayudarán a reducir el peso del CSS generado. Consiste en pasarle un minimizador de CSS, que compacte el código, elimine comentarios y cosas como esas.

Esto lo podemos conseguir mediante un plugin de PostCSS llamado "cssnano". Para usarlo tienes que comenzar instalando el plugin en el proyecto con npm.

npm i -D cssnano

Luego lo configuramos como un plugin más dentro del archivo postcss.config.js.

const purgecss = require('@fullhuman/postcss-purgecss');

module.exports = {

plugins: [

require('tailwindcss'),

require('autoprefixer'),

purgecss({

content: [

'./dist/\*\*/\*.html'

]}),

require('cssnano')({

preset: 'default',

}),

]

}

En este caso estamos usando el preset de cssnamo llamado "default", que será el ideal en la mayoría de los casos.

Las reducciones obtenidas por medio de la minimificación pueden ser relevantes, a veces mayores del 50% del peso del archivo, aunque depende lógicamente cómo se ha codificado el CSS original.

## **Plugin para anidación de reglas CSS**

En PostCSS tenemos la posibilidad de usar dos plugin distintos para anidación de reglas de CSS. Eligiremos uno u otro en función de la sintaxis que queramos usar.

* PostCSS Nesting: Este soporta la sintaxis de la especificación de CSS Nesting
* PostCSS Nested: En este caso soporta la sintaxis de anidación de CSS implementada en Sass

## **PostCSS Nested**

Por ejemplo, una vez instalado PostCSS en el proyecto, para usar PostCSS Nested realizaremos el siguiente comando de instalación:

npm install -D postcss-nested

Configuramos el postcss.config.js con el siguiente código:

module.exports = {

plugins: [

require('postcss-nested'),

]

}

A continuación podremos hacer anidación de reglas CSS, en este caso con la sintaxis de Sass. Sirva este código de muestra:

h1 {

color: red;

&.main {

color: green;

font-size: 2.5rem;

}

}

PostCSS Nesting

Si preferimos usar la especificación de CSS Nesting podemos usar el plugin PostCSS Nesting, que instalamos con el comando:

npm install -D postcss-nesting

Luego se incluye el plugin en la configuración de PostCSS dentro del archivo postcss.config.js.

module.exports = {

plugins: [

require('postcss-nesting'),

]

}