Transpilación de código – babel

Consiste en convertir nuestro código a código valido para subir a un servidor. Todo lo que hemos escrito hasta ahora es código de desarrollo, ese código si lo ejecutamos en un navegador nuevo va a funcionar sin ningún problema, pero si subimos ese código a un servidor va a ver muchos navegadores que no reconozcan parte del código que estamos escribiendo por eso tenemos que convertir ese código a lo que se denomina código de producción es decir el código que va a ver el usuario para ello vamos a seguir una serie de pasos.

Vamos a aprender la forma de reutilizar esto que vamos a hacer para todos los proyectos para que solo lo tengamos que escribir una vez y después lo podamos reutilizar si ningún tipo de problema

Instalación de node

Comprobamos que no tengamos ninguna versión de **node** instalada en agregar o quitar programas si la tenemos lo eliminamos.

Descargamos **node** desde la página oficial

Al terminar de instalar abrimos un **cmd**

Escribimos el comando **node --version**

Si al ejecutar el comando no sale la versión de node lo que tenemos que hacer es reiniciar el equipo y volver a ejecutar el comando.

Ejecutamos **npm --version**

Abrir una terminal

Creamos dos carpetas **dev** y **public** dentro de cada una de ellas una carpeta **js** en **/dev/js/** será donde tengamos el código de desarrollo donde vamos a trabajar y en **/public/js/** será donde se almacene nuestro código transpilado a ecmascript 5 donde no se suele tocar nada y es la que se sube al servidor.

visualStudioCode -> terminal -> nueva terminal

La ventaja de hacerlo desde visualStudioCode es que no necesitamos escribir la ruta donde se almacena el proyecto ya nos la autocompleta el mismo.

Iniciamos node en el proyecto con el comando: **npm init -y** esto crea un archivo **package.json** una vez tengamos el archivo copiamos y pegamos este comando para instalar varios paquetes que necesitamos para convertir el código javascript 2019 a ecma script 5 la versión que soporta todos los navegadores por muy viejos que sean.

npm install @babel/cli @babel/core @babel/polyfill @babel/preset-env @babel/register gulp gulp-babel gulp-concat gulp-plumber gulp-uglify --save-dev

Una vez terminado nos dejara una carpeta llamada **node_modules** que viene a ser el núcleo de node más todos los paquetes que acabamos de instalar y el **package-lock.json** este archivo no

lo tenemos que tocar lo que tiene es el historial de lo que ha hecho node con todo lo que ha ido instalando de donde lo ha instalado etc este archivo no se toca porque es posible que dejen de funcionar nuestro transpilador.

Creamos un nuevo archivo **gulpfile.babel.js** este va a ser nuestro archivo de configuración para galp, galp es un automatizador de tareas que lo que va a hacer es convertir nuestro código de código nuevo a código viejo cada vez que nosotros guardemos.

Pegamos el siguiente código en el archivo **gulpfile.babel.js**

```
import gulp from "gulp"
import babel from "gulp-babel"
import concat from "gulp-concat"
import uglify from "gulp-uglify"
import plumber from "gulp-plumber"

gulp.task("babel", () => {
    return gulp
        .src("dev/js/*.js")
        .pipe(plumber())
        .pipe(
            babel({
                presets: ["@babel/preset-env"],
            }),
        )
        .pipe(concat("scripts-min.js"))
        .pipe(uglify())
        .pipe(gulp.dest("public/js"))
})

gulp.task("default", () => {
    gulp.watch("dev/js/*.js", gulp.series("babel"))
})
```

import gulp from "gulp"

import babel from "gulp-babel"

import concat from "gulp-concat"

import uglify from "gulp-uglify"

import plumber from "gulp-plumber"

gulp.task("babel", () => {

  return gulp

    .src("dev/js/*.js")

    .pipe(plumber())

    .pipe(

      babel({

        presets: ["@babel/preset-env"],

      }),

```
    )
    .pipe(concat("scripts-min.js"))
    .pipe(uglify())
    .pipe(gulp.dest("public/js"))
})

gulp.task("default", () => {
  gulp.watch("dev/js/*.js", gulp.series("babel"))
})
```

Creamos el archivo: **.babelrc** este es el archivo de configuración de babel es un json que tiene el presset o librería que va a utilizar para convertir el código lo único que necesita es esto:


```
{
  "presets": [
    "@babel/preset-env"
  ]
}
```

Instalamos gulp en todos los usuarios con **npm install -g gulp** esto se instala solo una vez porque es a nivel de todo el equipo.

Una vez hecho esto ya podemos empezar a convertir código. Si escribimos en la terminal **galp** se queda gulp en modo escucha cada vez que escribamos código se va a convertir automáticamente en **/public/js/**

Como reutilizar todo el código para futuros proyectos

Necesitamos la misma estructura de carpetas de **dev** y **public** si queremos cambiar las rutas en el archivo gulpfile:

Copiamos los archivos: **gulpfile.babel.js package.json** y **.babelrc** los pegamos en la carpeta del nuevo proyecto y ejecutamos **npm install** este comando lo que va a hacer es leer el archivo **package.json** e instalar todas las dependencias que necesite sin tener que hacer nada más y volver a ejecutar **gulp**