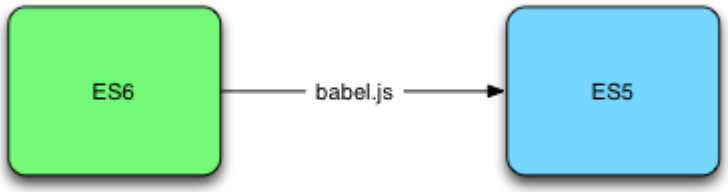
Babel.js es un transcompilador que nos permite convertir nuestro código de JavaScript ES6 en código de ES5. Esta característica se esta convirtiendo en algo crítico para mucha gente ya que las nuevas características de ES6 hacen deseable trabajar con el lenguaje lo antes posible. Lamentablemente hoy en día en la mayor parte de los navegadores y distintas plataformas de JavaScript tienen un soporte parcial de ES6 . Vamos a introducir Babel.js y como nos ayuda a transformar nuestro código de ES6 a ES5.



Babel.js y Node.js

Existen varias formas de utilizar Babel nosotros nos vamos ha apoyar en la versión de linea de comandos que realiza una compilación directa, para ello usamos Node.js e instalados el cliente de Babel.js.

npm install babel-cli

npm install babel-preset-es2015

Hemos instalado dos módulos , el primero es el propio cliente de Babel y el segundo es la configuración para que soporte JavaScript ES6 (ES 2015). El último paso que nos queda de realizar a nivel de configuración es crear el fichero de configuración de Babel ".babelrc" y configurarlo para que quede configurado por defecto para ES 2015 .

{ "presets": ["es2015"] }

Es suficiente con añadir una única línea. Es momento de comenzar a usar el cliente, en este

caso vamos a partir de un fichero denominado origen.js que contiene el siguiente código:

```
var lista=[2,3,5,7];

lista.map(x=> x*x).forEach(x=>console.log(x));
```

Este código pertenece a ES6 ya que usa las nuevas arrow functions y queremos que babel lo convierta a ES5 , para ello utilizamos la linea de comandos:

babel origen.js –out-file destino.js

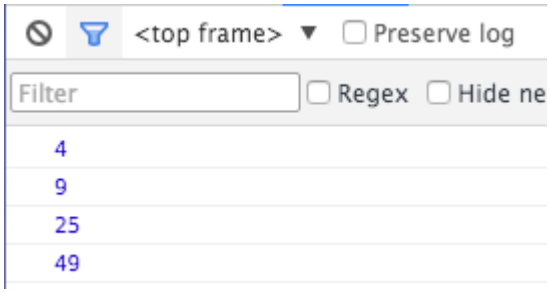
A continuación podemos ver el fichero destino.js que Babel.js nos genera:

```
"use strict";

var lista = [2, 3, 5, 7];

lista.map(function (x) {
 return x * x;
}).forEach(function (x) {
 return console.log(x);
});
```

Este código es claramente reconocible por cualquiera que trabaje con JavaScript clásico (ES5) . Lo cargamos en un navegador y veremos como imprime por consola los cuadrados de los números que se encuentran en el array.



Podemos configurar Babel para que cada vez que modifiquemos el fichero origen actualice automaticamente el fichero de destino a través de un watcher.

babel origen.js –watch –out-file destino.js

Cada día usaremos más este transcompilador ya que también nos permite operar con React.js de una forma mucho más cómoda y su sistema de plugins le hace muy extensible.