React.JS

Introducción

Módulo 01

Herramientas

Transpiladores

Los transpiladores son programas capaces de traducir el código de un lenguaje para otro, o de una versión para otra. Por ejemplo, el código escrito en ES6, traducirlo a ES5. Dicho de otra manera, el código con posibles problemas de compatibilidad, hacerlo compatible con cualquier plataforma 5.

El transpilador es una herramienta que se usa durante la fase de desarrollo. En esa fase el programador escribe el código y el transpilador lo convierte en un proceso de "tradución/compilación = transpilación". El código transpilado, compatible, es el que realmente se distribuye o se despliega para llevar a producción. Por tanto, todo el trabajo de traducción del código se queda solo en la etapa de desarrollo y no supone una carga mayor para el sistema donde se va a ejecutar de cara al público.

Transpiladores

Hoy tenemos transpiladores para traducir ES6 a ES5, pero también los hay para traducir de otros lenguajes a Javascript.

Quizás hayas oído hablar de TypeScript, o de CoffeeScript o Flow. Son lenguajes que una vez transpilados se convierten en Javascript ES5, compatible con cualquier plataforma.

Babel

Babel es una herramienta que nos permite transformar nuestro código JS de última generación (o con funcionalidades extras) a JS que cualquier navegador o versión de Node.js entienda.

Babel funciona mediante plugins para que le indiquemos qué cosas queremos que transforme, por ejemplo con el plugin *babel-plugin-transform-es2015-arrow-functions* podemos decirle que transforme las arrow functions de ECMAScript 2015 a funciones normales, con babel-plugin-transform-react-jsx podemos hacer que entienda código de JSX y lo convierta a código JS normal 6.

Webpack

Webpack se define como un empaquetador de módulos (un bundler en la jerga habitual) pero que hace muchísimas cosas más:

- Gestión de dependencias
- Ejecución de tareas
- Conversión de formatos
- Servidor de desarrollo
- Carga y uso de módulos de todo tipo (AMD, CommonJS o ES2015)

Y esto esto último lo que hace que destaque en especial. Es una herramienta extremadamente útil cuando desarrollamos aplicaciones web diseñadas con filosofía modular, es decir, separando el código en módulos que luego se utilizan como dependencias en otros módulos.

Una de las cosas que hace realmente bien Webpack es la gestión de esos módulos y de sus dependencias, pero también puede usarse para cuestiones como concatenación de código, minimización y ofuscación, verificación de buenas prácticas(linting), carga bajo demanda de módulos, etc.

Una de las muchas cosas interesantes de Webpack es que no solo el código JavaScript se considera un módulo. Las hojas de estilo, las páginas HTML e incluso las imágenes se pueden utilizar también como tales, lo cual da un extra de potencia muy interesante.

Webpack entonces es una herramienta de compilación (una build tool en la jerga) que coloca en un grafo de dependencias a todos los elementos que forman parte de tu proyecto de desarrollo: código JavaScript, HTML, CSS, plantillas, imágenes, fuentes.

Instalación

Dado que vamos a estar usando **npm** como manager de paquetes para controlar el desarrollo de la aplicación, podemos instalar webpack y babel simplemente con comandos de instalación de **npm** :

> npm install --save webpack webpack-cli webpack-dev-server
> npm install --save babel-loader babel-core babel-preset-es2015 babel-preset-react

Las librerías de *babel loader* y *babel core* son las que necesitamos para que webpack pueda integrar la transpilación del código y las de *preset* son para que babel sepa cómo interpretar y compilar nuestro código jsx en Javascript plano.

Puedes descargar Node.Js junto con NPM en https://nodejs.org/es/

Configuración

Lo próximo que necesitamos hacer es un archivo de configuración para cada paquete. Respectivamente ambos usan :

- webpack.config.js
- .babelrc

Estos dos archivos deben encontrarse en nuestro directorio raíz donde se encuentre el resto de la aplicación.

Más información en https://babeljs.io/

Dentro del archivo de configuración de Webpack, entre otras opciones podemos encontrar:

- **Entry:** el punto de entrada de nuestro programa. A partir de este archivo es por donde webpack va a comenzar a construir el bundle.
- **Output:** El punto de salida. Acá podemos especificar un directorio y nombre de archivo en donde se van a volcar todos los contenidos del bundle.
- **Module:** configuraciones que aplican para cada módulo, es decir, como babel y webpack van a compilar cada uno de los archivos que le digamos.
- **Mode:** la forma en que webpack va a compilar nuestros archivos.

¡Muchas gracias!

¡Sigamos trabajando!