Webpack

Gestores de dependencias: npm

Viene incluido con node.js.

Almacena la descripción y las dependencias del paquete en un archivo package.json.

npm init: crea un package.json

npm install jquery --save: instala y agrega la dependencia a package.json

npm install: instala todas las dependencias en package.json

Objetivos

Evitamos...

Escribir código que utilice variables globales.

Comprobar en la documentación qué dependencias deben cargarse primero.

Asegurar el orden de carga correcto a través del orden de las etiquetas <script>

Envolver el código en un IIFE

Queremos...

Utilizar un sistema de módulos estandarizado para dependencias y exportaciones.

Declarar dependencias explícitamente en cada archivo.

Utilice un módulo loader o bundler que entienda el sistema de módulos.

Module loaders y module bundlers

Module loader

- Se ejecuta en el navegador y carga los módulos cuando se solicitan.
- Es fácil de usar
- Menos optimizado para uso de producción.
- Ejemplos: requirejs, systemjs

Module bundler

- Se ejecuta en la compilación y empaqueta módulos en archivos estáticos
- Necesita un paso de preparación / construcción
- Es más optimizado para uso de producción
- Ejemplos: ***webpack***, browserify

Tareas de un module bundler

Lee entry point

Procesa los módulos especificados

Descarga las dependencias especificadas en los módulos

Lee, analiza, y procesa las dependencias

Agrupa todos los módulos usados

Agrupa los módulos para el proceso de ejecución de la aplicación

Saca el archivo empaquetado

Webpack: conceptos fundamentales

Descargar toda la aplicación web a la vez puede ser malo. Una webapp puede tener 10MB entre assets y código. La división de código divide la aplicación web en partes (chunks).

Webpack se basa en dos principios:

1. La segmentación de código está integrada en todos los aspectos del webpack.
2. Todo es un módulo: JS, CSS, HTML, imágenes...

La segmentación de código (code splitting)

Cada módulo tiene dependencias.

1. Construir un esquema de módulos conectados a través de dependencias. Hay dependencias de sincronización y asíncronas.
2. Construir un esquema de chunks: para cada punto de entrada, crea un punto de entrada (entry point).
3. Optimizar el esquema de chunks. Fusionar chunks con los mismos módulos.

Todo es un módulo

webpack permite configurar "loaders" para los archivos. Los loaders transforman archivos en módulos javascript.

Ejemplo: imagenes

CSS está en línea como una cadena en javascript. Cuando se importa, se agrega una etiqueta <style> al DOM.

Otros ejemplos: TypeScript, ES6, HTML...

Características de webpack

- Compilación incremental.
- Alto rendimiento a través de múltiples capas de caché de compilación.
- Compilación automática.
- Servidor de desarrollo.
- Reemplazo de módulos en tiempo de compilación.
- Configurable y extensible.
- Soporte de módulos AMD, CommonJs y ES6.
- Múltiples targets: web, node.js, electron, webWorker, etc.
- Múltiples puntos de entrada.
- Plugins de personalización.
- Generación de hashes para cada asset generado.

Enlaces

Webpack

Github de webpack