hillel IT school

Lesson 28

План заняття

- Webpack

Webpack

Сайт https://webpack.js.org/

Гітхаб https://github.com/webpack/webpack

Webpack — це збірник статичних модулів для сучасних програм JavaScript. Коли webpack обробляє вашу програму, він внутрішньо створює графік залежностей з однієї або кількох точок входу, а потім об'єднує кожен модуль, потрібний вашому проекту, в один або кілька пакетів, які є статичними активами для обслуговування вашого вмісту.

Core concepts

Entry - Точка входу вказує, який модуль webpack слід використовувати, щоб розпочати створення свого внутрішнього графіка залежностей. Webpack з'ясує, від яких інших модулів і бібліотек залежить ця точка входу (прямо чи опосередковано). За замовчуванням його значенням є ./src/index.js

Output - Властивість виводу повідомляє webpack, куди видавати пакети, які він створює, і як називати ці файли. За замовчуванням для основного вихідного файлу встановлено ./dist/main.js, а для будь-якого іншого згенерованого файлу — папку ./dist.

Loaders - З коробки webpack підтримує лише файли JavaScript і JSON. Loaders дозволяють webpack обробляти інші типи файлів і перетворювати їх на дійсні модулі, які можуть використовуватися вашою програмою та додаватися до графу залежностей.

Plugins - У той час як Loaders використовуються для перетворення певних типів модулів, плагіни можна використовувати для виконання ширшого спектру завдань, таких як оптимізація пакетів, керування активами та впровадження змінних середовища.

Mode - Встановивши для параметра mode значення development, production або none, ви можете ввімкнути вбудовану оптимізацію webpack, яка відповідає кожному середовищу. Значення за замовчуванням — production.

Докладніше тут https://webpack.js.org/concepts/

Instalations

Щоб запустити Webpack потрібно зробити нааступні команди:

1. npm init (якщо треба -y)
2. npm install webpack webpack-cli --save-dev
3. Додати index.html у папку прєку
4. Додати дерікторію src
5. Додати index.js до дерікторії src
6. Додати до package.json нову властивість - "private": true. Видалити "main": "index.js". Щоб запобігти випадковому паблішу коду.
7. Додати скріпт у package.json у розділ scripts => "build": "webpack"
8. Створити файл webpack.config.js у дерікторії проєкту та сконфігорувати його.

Configurations

Файл конфігурації webpack є файлом JavaScript, який експортує конфігурацію webpack. Ця конфігурація потім обробляється webpack на основі його визначених властивостей.

Оскільки це стандартний модуль Node.js CommonJS, ви можете робити наступне:

- Імпортуйте інші файли через вимогу (...)
- використовувати утиліти на npm через require(...)
- використовувати вирази потоку керування JavaScript, напр. оператор ?:
- використовуйте константи або змінні для часто використовуваних значень
- писати та виконувати функції для створення частини конфігурації
- Використовуйте ці функції, коли це доречно.

Усі опції конфігурацію можна знайти тут https://webpack.js.org/configuration/

Configurations

Базова конфігурація:

```
1   const path : PlatformPath | path  = require('path');
2
3   module.exports = {
4       mode: 'development',
5       entry: './foo.js',
6       output: {
7           path: path.resolve(__dirname, 'dist'),
8           filename: 'foo.bundle.js',
9       },
10  };
11
12
```

Webpack flags

Ви можете додати до скріптта деякі флаги

Які можете подивитись у терміналі за командою webpack – help

Наприклад webpack –mode=production запуститть webpack у прод моді.

Path

Стандартний nodejs модуль, який допомагає нам отримувати шлях до файлів у системі, на якій ми зараз працюємо.

Щоб їм скористатись, спочатку імпортуємо його const path = require('node:path'); Потім використовуємо за потребою.

__dirname та __filename є глобальними nodejs об’єктами які вказують на поточну дерікторію та файл

Методи:

- path.dirname(path) - Метод path.dirname() повертає назву каталогу шляху, подібно до команди Unix dirname.
- path.extname(path) - Метод path.extname() повертає розширення шляху з останнього входження
- path.join([...paths]) - Метод path.join() об’єднує всі надані сегменти шляху разом, використовуючи роздільник, що відповідає платформі, як роздільник, а потім нормалізує результуюче місце path.
- path.parse(path) - Метод path.parse() повертає об’єкт, властивості якого представляють важливі елементи шляху.
- path.resolve([...paths]) - Метод path.resolve() перетворює послідовність шляхів або сегментів шляху в абсолютний шлях.

Loaders

Повний список лодерів тут https://webpack.js.org/loaders/

Щоб використовувати лодери потрібно задати правила їх використання. Rules це об'єкти які складаються з:

- Rule.exclude - Виключити всі модулі, які відповідають будь-якій із цих умов.
- Rule.include - Включайте всі модулі, які відповідають будь-якій із цих умов.
- Rule.loader - це ярлик Rule.use: [{ loader }].
- Rule.loaders - є псевдонімом Rule.use.
- Rule.options / Rule.query - ярлики для Rule.use: [{ options }]
- Rule.use - може бути масивом UseEntry (https://webpack.js.org/configuration/module/#useentry), які застосовуються до модулів. Кожен запис визначає завантажувач, який буде використано.

Докладніше тут https://webpack.js.org/configuration/module/#modulerules

Babel

Підключимо можливість використовувати сучасний синтаксис js.

Для цього робимо декілька кроків:

- npm install -D babel-loader @babel/core @babel/preset-env
- Додаємо конфіг до webpack.config.js
- Та створюємо імпорт CSS до app.js

Config складається з:

- test - що буде шукати webpack, по типу RegExp
- exclude - що не попадає до аналізу та обробки.
- use - що буде використовувати. Це строка або масив з лодерів. Якщо ви використовуэте масив, то передавйте туди назву лодера або об’экт з його конфігурацією.

Детальніше тут https://webpack.js.org/loaders/babel-loader/

Список плагінів для babel https://babeljs.io/docs/plugins-list

Styles

Підключимо можливість використовувати CSS. Webpack орієнтован на js тому він буде очікувати підключення стилів у js файлу. Наприклад main.css імпортується у main.js

Для цього робимо декілька кроків:

- npm install --save-dev css-loader style-loder
- Додаємо конфіг до webpack.config.js
- Та створюємо імпорт CSS до app.js

Детальніше тут https://webpack.js.org/loaders/css-loader/

Styles

Підключимо можливість використовувати SASS та SCSS.

Для цього робимо декілька кроків:

- npm install sass-loader sass webpack --save-dev
- Додаємо конфіг до webpack.config.js
- Та створюємо імпорт SCSS або SASS до app.js

Детальніше тут https://webpack.js.org/loaders/sass-loader/

Html

Плагін HtmlWebpackPlugin спрощує створення файлів HTML для обслуговування пакетів webpack. Це особливо корисно для пакетів webpack, які містять хеш у назві файлу, який змінює кожну компіляцію. Ви можете або дозволити плагіну згенерувати для вас файл HTML, надати власний шаблон за допомогою шаблонів lodash або використати власний завантажувач.

Npm - https://www.npmjs.com/package/html-webpack-plugin

Webpack - https://webpack.js.org/plugins/html-webpack-plugin/

Щоб його запустити потрібно:

- Імпортувати його до webpacj.config.js
- створити новий інстанс у масиві plugins допомогою new HtmlWebpackPlugin

Він за дефолтом створить Index.html у папці output та додасть до нього скрипт file output.

HtmlWebpackPlugin options

hillel IT school

- Title - Заголовок для створеного документа HTML
- Filename - Файл для запису HTML. За замовчуванням index.html. Тут також можна вказати підкаталог (наприклад: активи/admin.html).
- ScriptLoading - Сучасні браузери підтримують неблокуюче завантаження JavaScript («defer») для покращення продуктивності запуску сторінки.
- Minify - Контролює, чи слід мінімізувати вихідні дані та якими способами.
- Hash - Якщо значення true, до всіх включених сценаріїв і файлів CSS додається унікальний хеш компіляції webpack
- Template - відносний або абсолютний шлях до шаблону. За замовчуванням він використовуватиме src/index.ejs, якщо він існує.

ImageMinimizerWebpackPlugin

Для оптимізації зображань потрібно інсталювати новий плагін

npm install image-minimizer-webpack-plugin imagemin --save-dev

Та лодери

npm install imagemin-gifsicle imagemin-mozjpeg imagemin-pngquant imagemin-svgo --save-dev

Далі додаємо конфігурацію

Докладніше тут https://webpack.js.org/plugins/image-minimizer-webpack-plugin/#getting-started

Тепер ми можемо шмпортувати зображення прямо до js файлів

ImageMinimizerWebpackPlugin

Для оптимізації зображань потрібно інсталювати новий плагін

npm install image-minimizer-webpack-plugin imagemin --save-dev

Та лодери

npm install imagemin-gifsicle imagemin-mozjpeg imagemin-pngquant imagemin-svgo --save-dev

Далі додаємо конфігурацію

Докладніше тут https://webpack.js.org/plugins/image-minimizer-webpack-plugin/#getting-started

Тепер ми можемо шмпортувати зображення прямо до js файлів

CopyWebpackPlugin

Щоб копіювати статичні файли потрібно використовувати новий плагін

https://webpack.js.org/plugins/copy-webpack-plugin/

Далі конфігуруємо його, та вказуємо звідки і куди клонувати фали

devServer

Для того щоб скористуватись можливостями відстежування змін у файлах потрібно встановити watch mod

Для цього потрідно запускати webpack --watch

Далі додаємо дев сервер. Webpack-dev-server надає вам елементарний веб-сервер і можливість використовувати живе перезавантаження.

Для цього:

- Інсталюємо сервер npm install --save-dev webpack-dev-server
- Далі додаємо конфіги.
- Та додаємо команду serve до webpack у package.json

Докладніше про devServer тут https://webpack.js.org/configuration/dev-server/#root

Докладніше про створення сервера тут https://webpack.js.org/api/webpack-dev-server/

Eslint

Для того щоб писати код чисто, та відстежувати помилки на етапі розробки гарною практикою є використання Eslint https://eslint.org/. Він допомагає усім розробникам писати в одному стилі.

ESLint статично аналізує ваш код, щоб швидко знаходити проблеми. Він вбудований у більшість текстових редакторів, і ви можете запускати ESLint як частину конвеєра безперервної інтеграції.

Документація тут https://eslint.org/docs/latest/

Щоб встановити Eslint портібно:

- Встановити пакети npm install eslint-webpack-plugin eslint --save-dev
- Додати плагін до списку new ESLintPlugin()
- Створити конфігурвційний файл для eslint. Ви можете зробити це власноруч або використати дефолтну команду npm init @eslint/config

Варіанти конфіг файлів https://eslint.org/docs/latest/use/configure/configuration-files

Варіанти правил https://eslint.org/docs/latest/rules/

Як налаштувати правила https://eslint.org/docs/latest/use/configure/rules

Сторонні бібліотеки

Для прикладу завантажимо бібліотеку bootstrap https://www.npmjs.com/package/bootstrap

Для підключення додаємо bootstrap у app.js

Докладніше тут https://getbootstrap.com/docs/4.0/getting-started/webpack/

Давайте завантажимо також lodash https://lodash.com/

Для цього завантиажимо бібліотеку npm і lodash з https://www.npmjs.com/package/lodash

Та підключимо її у файлі де нам це потрібно

Дякую за увагу