Java Script

NEXT

GENIUS space

План

Складання проектів: WebPack

робота з бібліотеками

JSON

Webpack - це збирач JS-модулів, менеджер модульних залежностей, який аналізує дерево залежностей та створює один або кілька результуючих файлів, що містять всю кодову базу проекту. Вибудовує порядок підключення модулів, збирає, мініфікує, запаковує та багато іншого.

Webpack став одним із найважливіших інструментів веб-розробника. В першу чергу це менеджер модульних залежностей програми та збирач JS-файлів, але він може трансформувати всі ресурси — HTML та CSS, SASS, LESS та PostCSS, оптимізувати зображення, компілювати шаблони, запускати локальний веб-сервер для розробки та багато іншого.

Які проблеми вирішує Webpack?

Зазвичай, при створенні програми на JavaScript код розділяється на кілька частин (модулів). Потім у файлі "index.html" необхідно вказати посилання на кожен скрипт.

```
<body>
<script src="http://cdnjs.cloudflare.com/ajax/libs/jquery/2.1.4/jquery.min.js"></script>
<script src="libs/react.min.js"></script>
<script src='src/admin.js'></script>
<script src='src/dashboard.js'></script>
<script src='src/api.js'></script>
<script src='src/auth.js'></script>
<script src='src/rickastley.js'></script>
</body>
```

Це не тільки втомлює, а й призводить до помилок.
Можливо не тільки не забути про якийсь скрипт, але й розташувати їх у правильному порядку. Якщо завантажити скрипт, що залежить від React, до завантаження самого React, програма зламається. Вебпак вирішує ці завдання. Не слід турбуватися про послідовне включення всіх скриптів.

```
<body>
<script src='dist/bundle.js'></script>
</body>
```

Як ми швидко дізнаємося, збір модулів є лише одним із аспектів роботи веб-паку. При необхідності можна змусити веб-пакет здійснити деякі перетворення модулів перед їх додаванням в бандл. Наприклад, перетворення SASS/LESS на звичайний CSS, або сучасного JavaScript в ES5 для старих браузерів.

Встановлення веб-паку

Після ініціалізації проекту за допомогою npm для роботи вебпаку потрібно встановити два пакети — webpack і webpack-cli.

npm i webpack webpack-cli -D

Після встановлення зазначених пакетів, веб-пакет потрібно налаштувати. Для цього створюється файл webpack.config.js, що експортує об'єкт. Цей об'єкт містить параметри веб-паку.

Точка входу

Скільки б модулів не містило додаток, завжди є єдина точка входу. Цей модуль включає інші. Зазвичай таким файлом є index.js. Це може виглядати так:

index.js
imports about.js
imports dashboard.js
imports api.js

Якщо ми повідомимо веб-паку шлях до цього файлу, він використовує його для створення графа залежностей програми. Для цього необхідно додати властивість entry у налаштування вебпаку зі значенням шляху до головного файлу:

```
module.exports = {
 entry: './app/index.js'
}
```

JavaScript існує вже більше 20 років, і це одна з тих мов, які ніколи не перестають розвиватися. Мова має власну екосистему бібліотек, фреймворків, інструментів, менеджерів пакетів та нових мов, які компілюються у JavaScript.

Терміни фреймворк, бібліотека та інструмент можуть означати різні речі, залежно від контексту. Грань між ними досить невиразна. Фреймворк може містити бібліотеку. Бібліотека може реалізовувати фреймворкоподібні методи. Інструменти можуть бути невід'ємною частиною обох. Дамо загальні визначення.

Бібліотека — це структурований набір корисного функціоналу: функцій, об'єктів чи класів, які можна використовувати у своїй додатку. Бібліотека абстрагує різні шари, тому вам не потрібно турбуватися про їх деталі реалізації. Бібліотека скорочує час розробки, дозволяючи вам не турбуватися про дрібниці.

Бібліотека може містити функції для роботи з рядками, датами, DOM-елементами, подіями, анімацією, HTTP-запитами та багатьма іншими. Кожна функція може повертати значення коду, що викликає її, який може надалі використовувати їх залежно від логіки розробника.

Негативні сторони:

Баг у реалізації бібліотеки може викликати складнощі у його знаходженні та виправленні.
Немає гарантії, що команда розробників оперативно випустить патч.
Патч може змінити API, що спричинить значні зміни у вашому коді.

Фреймворк

Фреймворк - це каркас (скелет) програми. Він зобов'язує розробника будувати архітектуру програми відповідно до деякої логіки, а також надає деякий інтерфейс для взаємодії з собою. Фреймворк зазвичай надає функціонал на кшталт подій, сховищ та зв'язування даних, шаблониацію та інші.

За аналогією з машиною, фреймворк надає готові рами, корпус та двигун. Ви можете додавати, видаляти або замінювати деякі деталі, при цьому припускаючи, що автомобіль залишиться в робочому стані.

Фреймворк знаходиться на вищому рівні абстракції в порівнянні з бібліотекою і дозволить вам легко побудувати більшу частину програми.

Різниця між фреймворком і бібліотекою полягає в тому, що бібліотеку ви використовуєте у своєму коді, а у випадку з фреймворком ви пишете свій код в архітектурних конструкціях фреймворку.

Його мінуси:
Фреймворк може накладати обмеження, тому можуть бути проблеми з використанням функціоналу.
Велика екосистема пакетів може збивати з пантелику і вимагає часу на вивчення.

JSON

JSON – сучасний формат зберігання та передачі даних.
Об'єктоподібний синтаксис JSON дуже зручний. Саме в цьому форматі дані приходитимуть і відправлятимуться на сервер, зберігатимуться в локальному сховищі тощо.

JSON можна розглядати як рядкове представлення об'єкта JavaScript. Підтримує такі значення: string, number, object, array, boolean, null.

Javascript та JSON відмінно працюють разом завдяки методам JSON.stringify() та JSON.parse(). Вони перетворюють JS-об'єкт у JSON і назад, і незалежно від того, що у вас є, можна легко отримати зворотне.

Онлайн валідатор https://jsonlint.com/

Метод JSON.stringify(obj)

Щоб використовувати JSON.stringify(), просто передайте JavaScript-об'єкт, який потрібно перетворити на JSON.

```
const user = {
  name: 'Ivan',
  age: 30,
  city: "Lviv",
};

const dogJSON = JSON.stringify(user);
console.log(userJSON); // {"name":"Ivan","age":30,"city":"Lviv"}
```

JSON.stringify(dog) взяв об'єкт user і перетворив його на рядок. Цей рядок тепер валідний JSON і може бути збережений у файл або передано по мережі.

Метод JSON.parse(string)

Щоб отримати JavaScript-об'єкт із JSON, ви повинні розібрати (розпарсувати) його, операція зворотна stringify.

```
const json = '{"name":"Ivan","age":30,"city":"Lviv"}';

const user = JSON.parse(json);
console.log(user);
```

Тепер коли user є звичайним JavaScript-об'єктом, його можна використовувати.

Додаткові матеріали

- https://medium.com/a-beginners-guide-for-webpack-2/index-html-using-html-webpack-plugin-85eabdb73474
- https://www.learnhowtoprogram.com/intermediate-javascript/test-driven-development-and-environments-with-javascript/processing-html-with-a-webpack-plugin
- https://www.youtube.com/watch?v=WVEvhwv3cBs
- https://yonatankra.com/how-to-use-htmlwebpackplugin-for-multiple-entries/
- https://www.npmjs.com/package/html-webpack-plugin
- https://webpack.js.org/plugins/html-webpack-plugin/
- https://www.json.org/json-en.html
-

Домашнє завдання

Зробити збірку і додати до неї 2 бібліотеки
на Ваш розсуд