Java Script

NEXT

План

Local Storage

Node.js

додавання - видалення бібліотек

LocalStorage, sessionStorage

Об’єкти веб-сховища localStorage та sessionStorage дозволяють зберігати дані в браузері у вигляди пар ключ/значення.
Що цікаво, дані зберігаються навіть після оновлення сторінки (для sessionStorage) і після повного закриття і нового відкриття вікна браузера (для localStorage). Ми скоро перевіримо це на практиці.

Ми вже маємо cookies. Навіщо додаткові об’єкти?
На відміну від файлів cookies, об’єкти веб-сховища не надсилаються на сервер із кожним запитом. Завдяки цьому ми можемо зберігати набагато більше даних. Більшість браузерів дозволяють принаймні 2 мегабайти даних (або більше), користувач може навіть змінити цей об’єм.
Крім того, на відміну від файлів cookies, сервер не може маніпулювати об’єктами сховища через HTTP-заголовки. Все зроблено на JavaScript.
Сховище прив’язане до оригінального сайту (домен/протокол/порт). Таким чином, що різні протоколи або субдомени мають різні об’єкти зберігання, і не можуть отримати доступ до даних один одного.

Обидва об’єкти сховища забезпечують однакові методи та властивості:

setItem(key, value) – зберегти пару ключ/значення.

getItem(key) – отримати значення за ключем.

removeItem(key) – видалити дані за ключем.

clear() – видалити все.

key(index) – отримати ключ на заданій позиції.

length – кількість збережених елементів.

Як ви можете бачити, це схоже на колекцію Map (setItem/getItem/removeItem), але також дозволяє отримати доступ за індексом за допомогою key(index).

Зверніть увагу, що і ключ, і значення мають бути рядками.

Якщо ми спробуємо використати будь-який інший тип, наприклад число або об’єкт, він автоматично перетвориться на рядок:

localStorage.user = {name: "John"};
alert(localStorage.user); // [object Object]
Однак ми можемо використати JSON для зберігання об’єктів:

localStorage.user = JSON.stringify({name: "Тарас"});
// через деякий чаclet user = JSON.parse(localStorage.user);alert(user.name); // Тарас

Також можна перетворити весь об’єкт сховища на JSON рядок, наприклад, підчас налагодження коду:
// додано параметри форматування до JSON.stringify, щоб об’єкт виглядав кращеalert(JSON.stringify(localStorage, null, 2));

Node.js

Node.js — легке та ефективне середовище виконання JavaScript. Дозволяє писати високопродуктивні серверні програми та інструменти.

Спочатку Node.js створювався як серверне оточення для програм, але розробники почали використовувати його для створення інструментів, які допомагають автоматизувати виконання локальних завдань. У результаті нова екосистема інструментів (на кшталт Grunt і Gulp), що виникла навколо Node.js, призвела до трансформації процесу фронтенд-розробки.

Щоб встановити останню версію, перейдіть на офіційну сторінку, https://nodejs.org/en завантажте інсталятор і дотримуйтесь вказівок, достатньо просто натискати Next. Є інсталятори для Windows і MacOS, а також скомпіловані бінарники та вихідний код для Linux.
Після встановлення в терміналі буде доступна команда node. Для того щоб переконатися, що установка пройшла успішно, перевірте версію, запустивши в консолі команду node з прапором version.
node --version

Щоб використовувати всі інструменти (або пакети) Node.js, нам потрібна можливість встановлювати і керувати ними. Для цього створено NPM (node package manager) – пакетний менеджер Node.js. Він встановлює потрібні пакети та надає зручний інтерфейс для роботи з ними.
NPM складається з трьох основних компонентів:

Сайт npmjs.com - використовується для пошуку та ознайомлення з документацією пакетів.

Інтерфейс командного рядка (CLI) — запускається з терміналу та надає набір команд для роботи з реєстром та пакетами. Дозволяє створювати скрипти для запуску у терміналі.

Реєстр пакетів (registry) – велика загальнодоступна база даних інструментів розробки (пакетів).

Команди NPM

npm init — ініціалізує npm та створює файл package.json
npm install — встановлює всі залежності, перелічені в package.json
npm list --depth=0 - виведе у терміналі список локально встановлених пакетів із номерами їх версій, без залежностей
npm install [package-name] — установить пакет локально у папку node_modules
npm uninstall [package-name] — видалити пакет встановлений локально та оновитьpackage.json
npm start і npm test - запустить скрипт start або test, розташований в package.json
npm run [custom-script] - запустить кастомний скрипт, розташований у package.json
npm outdated — використовується для пошуку оновлень, виявить сумісні версії програмно та виведе список доступних оновлень
npm update — оновить усі пакети до максимально дозволеної версії

Ініціалізація проекту

Кожен проект починається зі створення файлу package.json - він відстежує залежності, містить службову інформацію, дозволяє писати npm-скрипти, і служить інструкцією при створенні нового проекту на основі готових налаштувань.

Створити файл package.json можна npm-командою init, тим самим ініціалізувавши проект у поточній папці.
npm init

Вам буде запропоновано ввести назву проекту, версію, опис тощо. Можна просто натискати Enter, доки не буде створено package.json і розміщено в папці проекту. Щоб не натискати Enter, пропускаючи порожні поля, використовується команда init із прапором --yes. Прапор — додаткове налаштування команди.
npm init --yes

Буде створено package.json із значеннями за замовчуванням. Щоб встановити ці значення, у терміналі послідовно виконайте такі команди, підставивши своє ім'я та пошту.

npm config set init.author.name "YOUR_NAME"
npm config set init.author.email "YOUR_EMAIL"

Можна редагувати файл package.json вручну або ще раз запустити npm init..

NPM-скрипти
Скрипти дозволяють запускати на виконання встановлені пакети.
Використовуючи npm-скрипти, можна створювати цілі системи складання проекту.

Автоматизуємо запуск index.js. Для цього у файлі package.json у полі scripts додамо скрипт запуску start.

```
{
"name": "node-test",
"version": "1.0.0",
"main": "index.js",
"scripts": {
"test": "echo \"Error: no test specified\" && exit 1";
"start": "node index.js"
},
"author": "k.malitowska <email@gmail.com>",
"license": "ISC",
"keywords": [],
"description": ""
}
```

Встановлення пакетів

Одна з можливостей, що надає npm, — установка пакетів, які витягуються з реєстру і розпаковуються в папку node_modules в корені проекту.

Після створення файлу package.json можна додати залежність до проекту. Залежністю називають npm-пакет, який використовується при розробці. Це всілякі утиліти та бібліотеки.

Встановимо бібліотеку validator.js для валідації рядків, наприклад, введення користувача в поля форми.

npm install validator

NPM завантажив validator і помістив його в node_modules - папку, в якій будуть всі зовнішні залежності.

Не додавайте папку node_modules до системи контролю версій, у всіх розробників вона буде своя. Якщо ви використовуєте Git, не забувайте додати папку node_modules до файлу .gitignore

Зверніть увагу на створений файл package-lock.json – це журнал знімків дерева залежностей проекту. Він гарантує, що команда розробників використовує ті самі версії залежностей. NPM автоматично оновлює його при додаванні, видаленні та оновленні пакетів.

У package.json з'явилася нова залежність у полі dependencies. Це означає, що validator версії 11.1.0 був встановлений як залежність та готовий до роботи. Пакети постійно оновлюються, версія може відрізнятись

Видалення пакетів

Припустимо, що встановлена у попередньому прикладі версія validator викликає проблеми із сумісністю. Ми можемо видалити цей пакет і поставити старішу версію.
npm uninstall validator

7. Встановлення певної версії пакета

Тепер встановимо необхідну версію validator. У команді встановлення номер версії вказується після символу @.

npm install validator@1.0.0

Установка пакетів певної версії використовується в комерчерських проектах, щоб гарантувати роботу кодової бази та можливість довгострокової підтримки.

Типи залежностей

У команд npm install та npm uninstall існують 3 прапори.

--save - Вказує що додається залежність яка увійде у фінальний продукт. Пакет буде встановлений локально, до папки node_modules, і буде додано запис у полі dependencies до package.json.

--save-dev - Вказує що додається залежність розробки. Пакет буде встановлено локально, до папки node_modules, і буде додано запис у поле devDependencies до package.json.

--global — вказує на те, що додається глобальна залежність, тобто інструмент який доступний для будь-якого проекту. Пакет буде встановлено глобально (до системи).

Керування версіями пакетів

Пакети мають пов'язаний із ними номер версії. Номери версій відповідають стандарту SemVer.

npm outdated – використовується для пошуку оновлень, виявить сумісні версії програмно.

npm update – оновить усі пакети до максимально дозволеної версії.

npm update [ім'я-пакета] - оновить вказаний пакет

Додаткові матеріали

- https://uk.javascript.info/localstorage
- http://xn--80adth0aefm3i.xn--j1amh/localstorage
- https://nodejs.org/en
- https://docs.npmjs.com/
- https://deliciousbrains.com/npm-build-script/

Домашнє завдання

Встановити Node.js стабільної версії