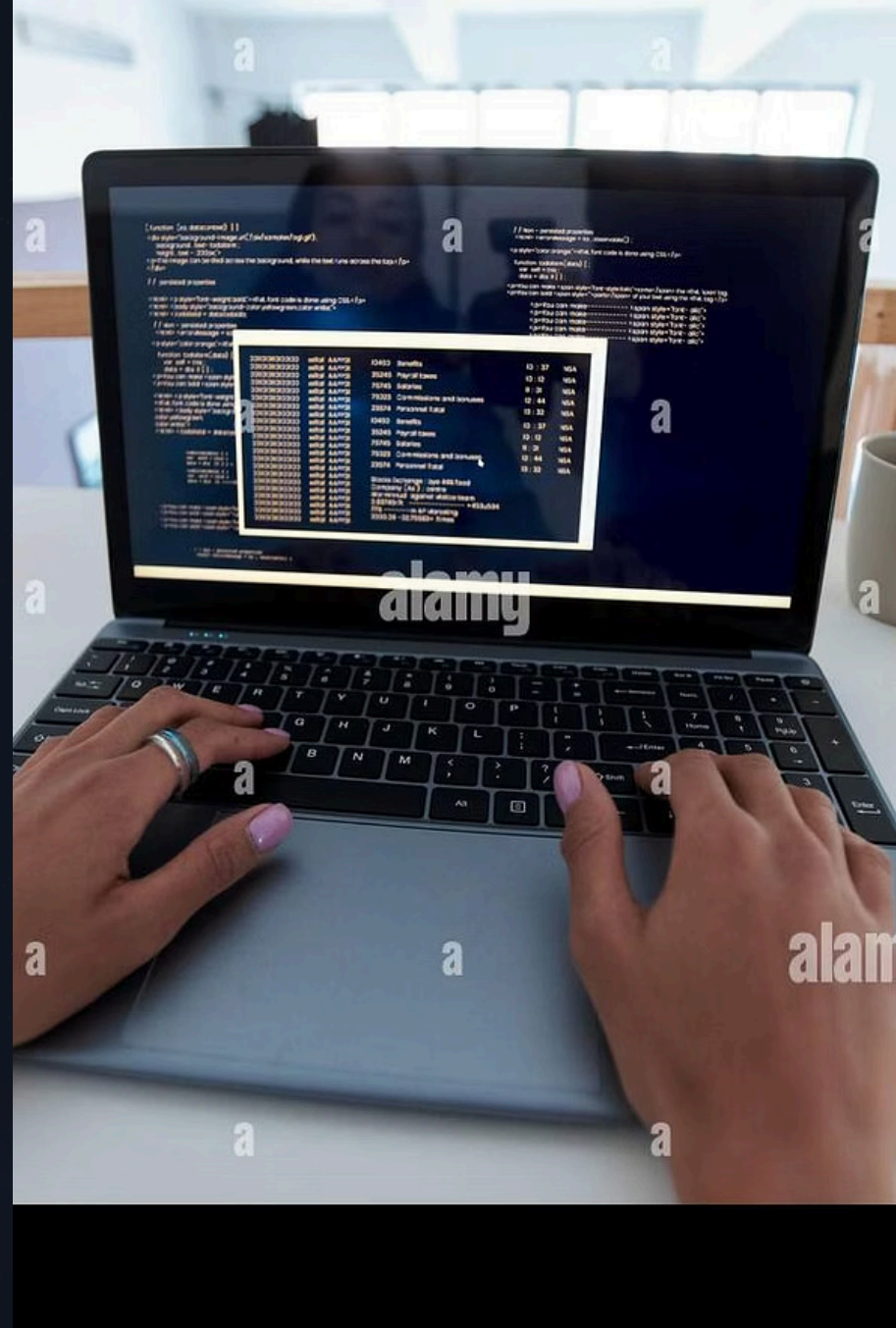
Використання Faker для Генерації Динамічних Даних в API Тестуванні

Автоматизація API-тестування потребує якісних тестових даних. Бібліотека Faker — це потужний інструмент для генерації реалістичних випадкових даних, який дозволяє створювати незалежні тести без необхідності ручного введення інформації.

Використання Faker забезпечує гнучкість та автономність ваших тестів, дозволяючи запускати їх багаторазово з унікальними даними кожного разу. Це особливо важливо для API-тестування, де потрібно перевіряти різні сценарії з різними вхідними параметрами.



Встановлення та Налаштування Faker

Встановлення пакету

Для початку роботи з Faker встановіть бібліотеку через npm як dev-залежність:

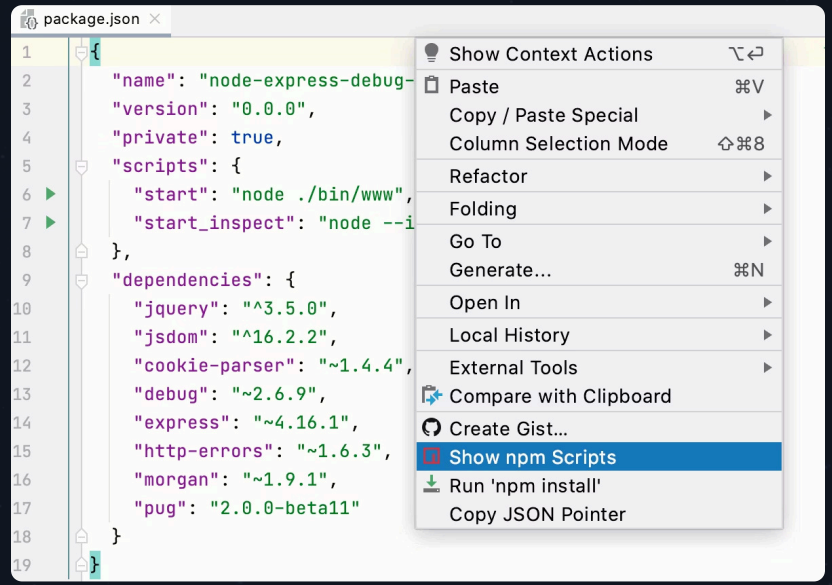
```
npm install --save-dev @faker-js/faker
```

Ця команда додає пакет до секції devDependencies у вашому файлі package.json, оскільки Faker використовується виключно для тестування та розробки.

Перевірка встановлення

Після успішного встановлення перевірте наявність пакету у файлі `package.json` в секції devDependencies. Ви побачите запис:

```
"@faker-js/faker": "^версія"
```



Важливо: Використовуйте flag `--save-dev`, щоб пакет не потрапив у production збірку вашого проєкту.

Базове Використання Faker у Запитах

Після встановлення імпортуйте бібліотеку у ваш тестовий файл та використовуйте методи генерації даних безпосередньо в API запитах.

Приклад створення користувача з динамічними даними

```
const axios = require('axios')
const {faker} = require('@faker-js/faker')
const jsonData = require('./env.json');

test('Create user', async() => {
  const createUser = await axios.post(`${jsonData.baseUrl}/users/add`, {
    'firstName': faker.internet.userName(),
    'lastName': faker.internet.lastName(),
    'age': 250,
  }, {
    headers: {
      'Content-Type': 'application/json',
      "Authorization": jsonData.token
    }
  })
  console.log(createUser.data)
})
```

faker.internet.userName()

Генерує випадкове ім'я користувача для тестування

faker.internet.lastName()

Створює реалістичне прізвище користувача

Автоматична генерація

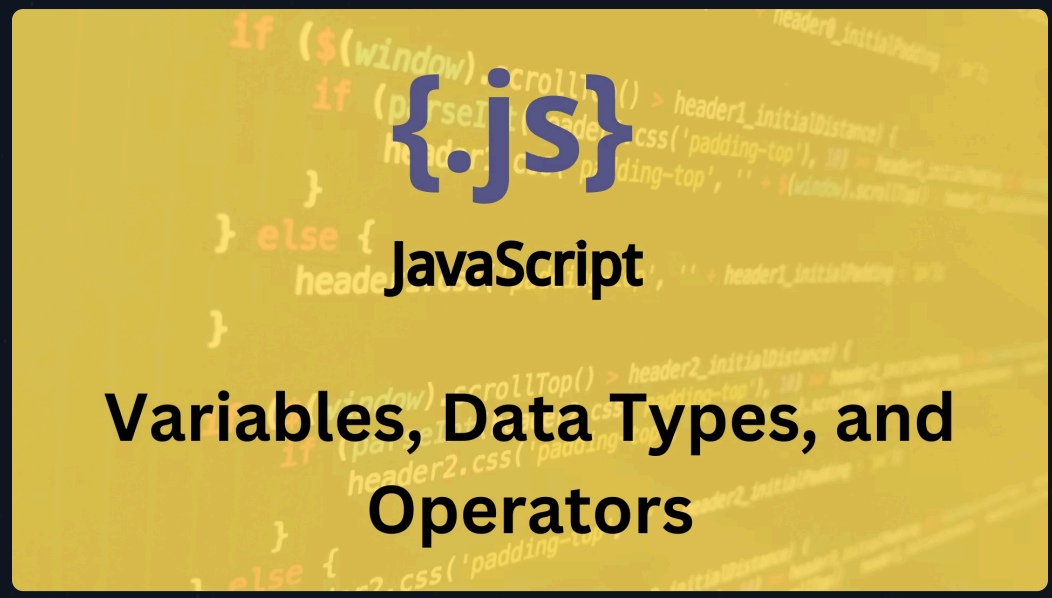
Кожен запуск тесту використовує нові унікальні дані

Збереження Згенерованих Даних у Змінних

Коли потрібен цей підхід?

Якщо вам потрібно використовувати ті самі згенороване дані в декількох місцях тесту або в різних тест-кейсах, зберігайте їх у змінних перед виконанням запитів.

Це забезпечує консистентність даних протягом усього тестового сценарію та дозволяє легко порівнювати результати.



Приклад з використанням змінних

```
const axios = require('axios')
const {faker} = require('@faker-js/faker')
const jsonData = require('./env.json');

let dataUserName = faker.internet.userName()
let dataUserLastName = faker.internet.lastName('male')

test('Create user', async() => {
  const createUser = await axios.post(`${jsonData.baseUrl}/users/add`, {
    'firstName': dataUserName,
    'lastName': dataUserLastName,
    'age': 250,
  }, {
    headers: {
      'Content-Type': 'application/json',
      "Authorization": jsonData.token
    }
  })
  console.log(createUser.data)
})
```

Переваги підходу: Змінні зберігають одні й ті самі дані протягом усього життєвого циклу тесту, що критично важливо для валідації відповідей API.

Перевикористання Даних для Валідації Відповідей

Найпотужніший сценарій використання Faker — це створення даних один раз та їх подальше використання для перевірки відповідей API. Цей підхід ідеально підходить для тестування CRUD операцій.

Повний приклад з валідацією

```
const axios = require('axios')
const {faker} = require('@faker-js/faker')
const jsonData = require('./env.json');

let dataUserName = faker.internet.userName()
let dataUserLastName = faker.internet.lastName('male')
let dataUserId = faker.string.alpha({ length: { max: 2}})

test('Create user', async() => {
  const createUser = await axios.post(`${jsonData.baseUrl}/users/add`, {
    'firstName': dataUserName,
    'lastName': dataUserLastName,
    'age': 250,
  }, {
    headers: {
      'Content-Type': 'application/json',
      "Authorization": jsonData.token
    }
  })
  console.log(createUser.data)
})

test('get user by id and compare values', async() => {
  const getUser = await axios.get(`${jsonData.baseUrl}/users/${dataUserId}`)

  expect(dataUserName).equal(getUser.data.firstName)
  expect(dataUserLastName).equal(getUser.data.lastName)
  expect(dataUserId).equal(getUser.data.id)
})
```

Генерація

Створюємо змінні з faker один раз

Використання

Відправляємо дані в POST запиті

Отримання

Запитуємо дані через GET запит

Валідація

Порівнюємо отримані дані зі згенерованими

Ключові переваги цього підходу

- Одноразова генерація даних
- Багаторазове використання в тестах
- Точна валідація відповідей API
- Незалежність тестів від зовнішніх даних