JS Generator 🎉

Agenda

1. Generator là gì?
2. Một vài ví dụ đơn giản về generator

1. Generator là gì?

- Generator là object được trả về từ generator function.
- Generator không thể được tạo ra trực tiếp.

Generator function

Cú pháp:

```
function* generateId() {
  yield 1;

  console.log('Continue to run');
  yield 2;

  console.log('Resume');
  return 3;
}

const newId = generateId();
newId.next(); // { value: 1, done: false }
newId.next(); // { value: 2, done: false }
newId.next(); // { value: 3, done: true }
```

Đặc điểm của một `generator function`:

- Hàm có thể dừng giữa chừng và tiếp tục sau.
- Mỗi lần gọi, hàm sẽ được thực thi cho đến khi gặp lệnh yield hoặc return.

Generator có 3 hàm:

Method	Description	Return
next()	Tiếp tục thực thi hàm cho đến khi gặp yield / return	`{ value: 2, done: true/false }`
return()	Dừng generator function và return kết quả.	`{ value: 3, done: true }`
throw()	Dừng generator function và trả về lỗi.	`{ value: undefined, done: true }`

2. Một vài ví dụ đơn giản về generator

Tạo ID tăng dần

```
// định nghĩa một generator function
function* generateId() {
  let i = 0;

  while (true) {
    i++;
    yield i;
  }
}

// khi gọi generator function, thì sẽ được trả về một generator object
// tức newId là một generator, nên ta có thể dùng hàm .next(), .return()
và .throw()
const newId = generateId();
console.log(newId.next()); // { value: 1, done: false }
console.log(newId.next()); // { value: 2, done: false }
console.log(newId.next()); // { value: 3, done: false }
```

Tạo dãy số Fibonaci

```
function* generateFibonaci() {
  let prev = 0;
  let curr = 1;

  while (true) {
    yield curr;

    // Calculate the next number
    const next = prev + curr;
    prev = curr;
    curr = next;
  }
}

const fibo = generateFibonaci();
console.log(fibo.next()); // { value: 1, done: false }
console.log(fibo.next()); // { value: 1, done: false }
console.log(fibo.next()); // { value: 2, done: false }
console.log(fibo.next()); // { value: 3, done: false }
console.log(fibo.next()); // { value: 5, done: false }
```

Tạo biến iterable

```
function* loopRange(from, to) {
  for (let i = from; i <= to; i++) {
    yield i;
  }

  return to + 1;
}

const range = loopRange(0, 10);
for (const i of range) {
  console.log(i);
}

// Log from 0 to 10
```

🌐 Link tham khảo

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Generator
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Iterators_and_Generators
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/function*
- https://javascript.info/generators
- https://dev.to/rfornal/use-cases-for-javascript-generators-1npc

---

Cảm ơn các bạn đã xem video của mình!
Nhớ like, share và subscribe để cho bạn bè cùng xem nhen 😉

❤️ Ủng hộ mình làm videos thì đóng góp qua MoMo/ZaloPay: `0901 309 729` nhé!

Kết nối với mình:

- 💻 Kênh Easy Frontend: https://youtube.com/easyfrontend
- 🎉 Fan page Facebook: https://www.facebook.com/learn.easyfrontend
- 📝 Nhóm trao đổi Facebook: https://www.facebook.com/groups/easyfrontend