

Passing Parameters to Iterables

using `yield*` expression to delegate to another iterable object

The `next` method of iterators can be used to pass a value that becomes the value of the previous `yield` statement.

```
let greetings = function *() {
  let name = yield 'Hi!';
  yield `Hello, ${ name }!`;
}

let greetingIterator = greetings();

console.log( greetingIterator.next() );
//> Object {value: "Hi!", done: false}

console.log( greetingIterator.next( 'Lewis' ) );
//> Object {value: "Hello, Lewis!", done: false}
```



The return value of a generator becomes the return value of a `yield *` expression.

```
let sumSequence = function *( num ) {
  let sum = 0;
  for ( let i = 1; i <= num; ++i ) {
    sum += i;
    yield i;
  }
  return sum;
}

let wrapSumSequence = function *( num ) {
  let sum = yield *sumSequence( num );
  yield `The sum is: ${ sum }.`;
}

for ( let elem of wrapSumSequence( 3 ) ) {
  console.log( elem );
}
```





Now, let's discuss some practical applications of generators.