Generator Functions

Marques Woodson
STAFF SOFTWARE ENGINEER

@mwq27 www.marqueswoodson.com

Overview

Generator functions

The ‘Yield’ keyword

What is yield delegation?

Early completion of a generator

Generator error handling

Generator Function

A function that can be paused and resumed at a later time, while having the ability to pass values to and from the function at each pause point.

Generator Function Syntax

```
function* gen() {...}

function *gen() {...}

function * gen() {...}

const obj = {
  *gen(params) {...}
}
```

Executing the generator function alone DOES NOT execute its containing code

Yield

Yield keyword signals the pause point of a generator function.

Possible Yield Actions

Send a value to the iterator

- yield ‘goes to iterator’;

Receive a value from the iterator

- const x = yield; //
- it.next(‘value for x’); //=> x is now ‘value for x’

Yield Expression Placement

```
var y = yield 3;
```

```
const arr = [yield 2,
  yield 3, yield 4];
```

```
if (yield 4 === 8 )
        {…}
```

Yield Delegation

Yield delegation essentially allows a host generator function to control the iteration of a different generator function.

Generator Functions Include Return and Throw

generatorExample.js

```
function* randomNumbers() {
  while(true) {
    yield Math.floor(Math.random() * 1000);
  }
}
const it = randomNumbers();
it.return(); // Valid
it.throw(); // Valid
```

No need to manually implement the ‘return’ and ‘throw’ methods

Early Completion

iterator.return()

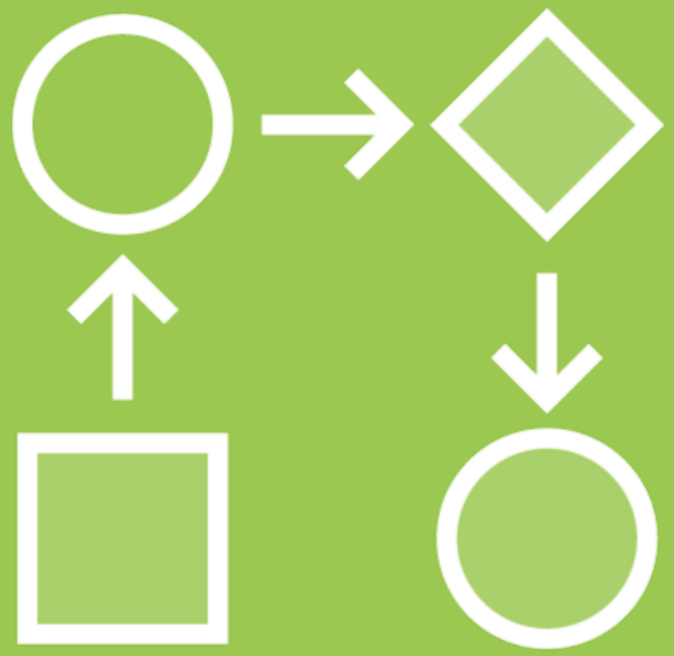
The return method ends a generator functions execution

iterator.throw()

The throw method will end a generator functions execution while also throwing an exception that can be handled by the generator

Summary

- Generator functions return iterators
- The ‘yield’ keyword
- Yield delegation (yield*)
- Early completion of generator
- Error handling



Next up: Introducing CAF

Cancelable Async Flows (CAF)

Make generator functions look like async functions