#### ES2019提案预览

##### 之生成器函数

摘要

JavaScript语言特性——生成器函数

@link <https://github.com/tc39/proposal-function.sent>

@link <https://github.com/tc39/proposal-generator-arrow-functions>

正文

什么是生成器函数？

Generator 函数是协程在 ES6 的实现，最大特点就是可以交出函数的执行权（即暂停执行）。它不同于普通函数，是可以暂停执行的，所以函数名之前要加星号，以示区别。整个 Generator 函数就是一个封装的异步任务，或者说是异步任务的容器。异步操作需要暂停的地方，都用 yield 语句注明。

调用一个生成器函数并不会马上执行它里面的语句，而是返回一个这个生成器的 迭代器 （iterator ）对象。当这个迭代器的 next() 方法被首次（后续）调用时，其内的语句会执行到第一个（后续）出现yield的位置为止，yield 后紧跟迭代器要返回的值。或者如果用的是 yield\*（多了个星号），则表示将执行权移交给另一个生成器函数（当前生成器暂停执行）。

生成器函数的应用场景——**长轮询**（Long Polling）

var fetch = require('node-fetch');

function\* gen(){

var url = 'https://api.github.com/users/github';

var result = yield fetch(url);

console.log(result.bio);

}

var g = gen();

var result = g.next();

result.value.then(function(data){

return data.json();

}).then(function(data){

g.next(data);

});

使用生成器函数实现长轮询与setInterval()/clearInterval()方式的不同在于：生成器函数在得到上一个请求的返回后，才会继续发出下一个请求。而setInterval()/clearInterval()定时发出一次请求，弱网络环境下会出现同时发出多个请求的情形，清除定时器的逻辑复杂。

Function.next

问题 Because there the first next call does not correspond to a yield within the generator function body there is currently no way for the code with the body to access the initial next argument.

解决 The value of function.sent within the body of a Generator Function is the value passed to the generator by the next method that most recently resumed execution of the generator. In particular, referencing function.sent prior to the first evaluation of a yield operator returns the argument value passed by the next call that started evaluation of the GeneratorBody.

□