js实践之生成指定个数（n）的指定整数范围[min,max]内的随机数组

今天在前端早读课看到这篇文章

感兴趣就自己试了一下，刚开始代码是如下：

```
function fn(n){
    var min = 2;
    var max = 32;
    // Let n be toUint32(lenValue).
    n = n >>> 0;
    console.log(' arr:',arr,' n:',n);

    var temp = [];
    var arr = new Array(n);
    arr.forEach(function(){
        let r = rd(min,max);
        console.info(arr.length, 'temp.includes(r):',temp.includes(r));
        while(temp.includes(r)){
            console.warn(' temp:',temp,' r:',r);
            r = rd(min,max);
        }
        temp.push(r);
    });
    return temp;
}

function rd(min,max){
   return Math.floor(Math.random()*(max-min+1)+min);
}
```

运行下面这行代码：
`fn(3)`

```
> fn(3)
  arr: undefined  n: 3
<- ▶ []
>
```

arr.forEach并没有执行。
查看forEach和new Array看到有下面2处描述：

Description

`forEach()` executes the provided `callback` once for each element present in the array in ascending order. It is not invoked for index properties that have been deleted or are uninitialized (i.e. on sparse arrays).

`callback` is invoked with **three arguments**:

- **the element value**
- **the element index**
- **the array being traversed**

Polyfill
Specifications
Browser compatibility
See also

原文：
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/forEach

Syntax

```
[element0, element1, ..., elementN]
new Array(element0, element1[, ...[, elementN]])
new Array(arrayLength)
```

Parameters

`elementN`
A JavaScript array is initialized with the given elements, except in the case where a single argument is passed to the `Array` constructor and that argument is a number (see the arrayLength parameter below). Note that this special case only applies to JavaScript arrays created with the `Array` constructor, not array literals created with the bracket syntax.

`arrayLength`
If the only argument passed to the `Array` constructor is an integer between 0 and 2^{32}-1 (inclusive), this returns a new JavaScript array with its `length` property set to that number (**Note:** this implies an array of `arrayLength` empty slots, not slots with actual `undefined` values). If the argument is any other number, a `RangeError` exception is thrown.

原文：
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array

即 new Array(num)，创建的是空位数组，即没有初始化的素组，forEach会跳过空位。 new Array(num)和[undefined]并不一样，可以看下面的例子：

```
> new Array()[0]
<- undefined
> undefined
<- undefined
> new Array(3)
<- ▶ [undefined × 3]
> 0 in  [undefined,undefined,undefined]
<- true
> 3 in  [undefined,undefined,undefined]
<- false
> new Array()[0] ===undefined
<- true
> |
```

从forEach的polyfill也可以看出，forEach过滤空位数组：

```
    }

    // 5. If thisArg was supplied, let T be thisArg; else let
    // T be undefined.
    if (arguments.length > 1) {
      T = arguments[1];
    }

    // 6. Let k be 0
    k = 0;

    // 7. Repeat, while k < len
    while (k < len) {

      var kValue;

      // a. Let Pk be ToString(k).
      //    This is implicit for LHS operands of the in operator
      // b. Let kPresent be the result of calling the HasProperty
      //    internal method of O with argument Pk.
      //    This step can be combined with c
      // c. If kPresent is true, then
      if (k in O) {

        // i. Let kValue be the result of calling the Get internal
        // method of O with argument Pk.
        kValue = O[k];

        // ii. Call the Call internal method of callback with T as
        // the this value and argument list containing kValue, k, and O.
        callback.call(T, kValue, k, O);
      }
      // d. Increase k by 1.
      k++;
    }
    // 8. return undefined
  };
}
```

Syntax
Parameters
Return value
Description
Examples
Printing the contents of an ar
Using `thisArg`
An object copy function
If the array is modified during elements might be skipped.
Polyfill
Specifications
Browser compatibility
See also

其实，阮一峰的es6入门，也有很详情的讲解：

8. 数组的空位

数组的空位指，数组的某一个位置没有任何值。比如， `Array` 构造函数返回的数组都是空位。

```
Array(3) // [, , ,]
```

上面代码中， `Array(3)` 返回一个具有3个空位的数组。

注意，空位不是 `undefined` ，一个位置的值等于 `undefined` ，依然是有值的。空位是没有任何值， `in` 运算符可以说明这一点。

```
0 in [undefined, undefined, undefined] // true
0 in [, , ,] // false
```

上面代码说明，第一个数组的0号位置是有值的，第二个数组的0号位置没有值。

ES5对空位的处理，已经很不一致了，大多数情况下会忽略空位。

- `forEach()` ， `filter()` ， `every()` 和 `some()` 都会跳过空位。
- `map()` 会跳过空位，但会保留这个值
- `join()` 和 `toString()` 会将空位视为 `undefined` ，而 `undefined` 和 `null` 会被处理成空字符串。

```
// forEach方法
[,'a'].forEach((x,i) => console.log(i)); // 1

// filter方法
['a',,'b'].filter(x => true) // ['a','b']

// every方法
[,'a'].every(x => x==='a') // true

// some方法
[,'a'].some(x => x !== 'a') // false

// map方法
[,'a'].map(x => 1) // [,1]
```

上一章 下一章

原文:数组的空位

接着把代码改成如下所示：

```
function fn(n){
    var min = 2;
    var max = 32;
    // Let n be toUint32(lenValue).
    n = n >>> 0;
    console.log(' arr:',arr,' n:',n);
    var temp = [];
    // var arr = new Array(n);
    var arr = [...new Array(n)];
    arr.forEach(function(){
        let r = rd(min,max);
        console.info(arr.length, 'temp.includes(r):',temp.includes(r));
        while(temp.includes(r)){
            console.warn(' temp:',temp,' r:',r);
            r = rd(min,max);
        }
        temp.push(r);
    });
    return temp;
}

function rd(min,max){
   return Math.floor(Math.random()*(max-min+1)+min);
}
```

但是上面fn还有个问题，就是现在并没有对n，进行判断，如果直接执行 `fn(100)` ,会直接死循环，因为temp的范围是【2，32】，无论如何都不可能输出100个【2，32]范围里的随机整数，结果在while里面死循环了。接着把代码改成如下所示：

```
function fn(n){
    var min = 2;
    var max = 32;
    //Let n be toUint32(lenValue).
    n = n >>> 0;
    if(n >= max){
        return [];
    }

    var temp = [];
    // var arr = new Array(n);
    var arr = [...new Array(n)];
    arr.forEach(function(){
        let r = rd(min,max);
        while(temp.includes(r)){
            r = rd(min,max);
        }
        temp.push(r);
    });
    return temp;
}

function rd(min,max){
   return Math.floor(Math.random()*(max-min+1)+min);
}
```

运行结果如下所示：

```
top ▼  Preserve log
  undefined
> console.time('b');cc=fn(32);console.info('cc:',cc);console.timeEnd('b')
  cc: ▶ []
  b: 2.050ms
<- undefined
> console.time('b');cc=fn(1);console.info('cc:',cc);console.timeEnd('b')
  cc: ▶ [5]
  b: 1.262ms
<- undefined
> console.time('b');cc=fn(-1);console.info('cc:',cc);console.timeEnd('b')
  cc: ▶ []
  b: 0.742ms
<- undefined
> console.time('b');cc=fn(31);console.info('cc:',cc);console.timeEnd('b')
  cc: ▶ [5, 9, 13, 11, 28, 20, 15, 32, 23, 24, 30, 31, 22, 21, 17, 3, 10, 2, 14, 6, 27, 4, 26, 25, 19, 12, 18, 8, 7, 16, 29]
  b: 2.040ms
<- undefined
> console.time('b');cc=fn(31);console.info('cc:',cc);console.timeEnd('b')
  cc: ▶ [32, 25, 7, 29, 24, 16, 30, 17, 5, 20, 28, 27, 2, 23, 11, 14, 13, 9, 15, 8, 21, 31, 12, 18, 19, 22, 3, 26, 10, 6, 4]
  b: 1.041ms
<- undefined
> console.time('b');cc=fn(31);console.info('cc:',cc);console.timeEnd('b')
  cc: ▶ [2, 23, 22, 3, 5, 25, 29, 27, 26, 31, 14, 13, 12, 10, 11, 15, 19, 21, 24, 7, 30, 28, 17, 32, 6, 8, 9, 20, 16, 4, 18]
  b: 0.975ms
<- undefined
> console.time('b');cc=fn(31);console.info('cc:',cc);console.timeEnd('b')
  cc: ▶ [11, 16, 8, 17, 23, 25, 29, 12, 14, 19, 27, 22, 31, 2, 26, 7, 10, 24, 32, 20, 15, 13, 30, 28, 6, 5, 9, 21, 4, 18, 3]
  b: 1.319ms
<- undefined
> console.time('b');cc=fn('a');console.info('cc:',cc);console.timeEnd('b')
  cc: ▶ []
  b: 0.765ms
<- undefined
>
```

多执行几次下面这行代码：

```
console.time('b');cc=fn(31);console.info('cc:',cc);console.timeEnd('b')
```

```
> console.time('b');cc=fn(31);console.info('cc:',cc);console.timeEnd('b')
  cc: ▶ [22, 28, 17, 7, 13, 10, 29, 20, 8, 25, 14, 27, 11, 24, 2, 31, 21, 9, 12, 16, 15, 3, 23, 18, 19, 30, 26, 4, 6, 5, 32]
  b: 20.584ms
<- undefined
```

从执行结果可以看出，当n取最大临界值的时候，有可能用的时间会比较久

参考链接：

1.MDN Array
2.MDN forEach
3.数组的空位