冷静想想，JS 是怎么输出对象的属性顺序的？

作者：Rabbitzzc
原文链接：https://ld246.com/article/1588781974097
来源网站：链滴

冷静想想，JS 是怎么输出对象的属性顺序的?

自从 es6 出来以后， Object.keys 方法得到了大量的使用，比如在需要遍历对象的属性与属性值进行独的样式输出等，那 Object.keys 是怎么实现的呢，它是如何保证对象的输出属性顺序的呢?

一般情况下，元素总是按属性赋值时的顺序（而不是按照名称顺序）排序的，可能会出现 bug，这时你的年终奖可能就保不住啦

上手实践

是骡子是马，先拿出来溜溜，首先需要了解 Object.keys 的实际效果。

round one

```
const a = {'1':'aaa','2':'bbb','3':'ccc','黑客':'ddd'}
Object.keys(a) //  ["1", "2", "3", "黑客"]
```

可以看到控制台打印为 ["1", "2", "3", "黑客"]。

round two

```
const a = {'黑客':'ddd','1':'aaa','2':'bbb','3':'ccc'}
Object.keys(a) //  ["1", "2", "3", "黑客"]
```

控制台打印为 ["1", "2", "3", "黑客"]，与表哥1打印结果是相同的。

round three

将名称由中文改为英文试试：

```
const a = {'ddd':'ddd','1':'aaa','2':'bbb','3':'ccc'}
Object.keys(a)
```

打印结果 ["1", "2", "3", "ddd"]。

从前面三位兄弟可以看到，似乎是根据 ACSII 码进行排序的，可以安心下结论了

但这是真的么?

round four

```
a = {'ddd':'ddd','1':'aaa','2':'bbb','3':'ccc', '黑客': 'ddd'}
Object.keys(a)
// ========
a = {'1':'aaa','2':'bbb','3':'ccc', '黑客': 'ddd','ddd':'ddd'}
Object.keys(a)
```

可以看到控制太打印的结果氛围为：

```
["1", "2", "3", "ddd", "黑客"]
```

["1", "2", "3", "黑客", "ddd"]

可以看到打印结果是不同的，因此可以根据反证法得出

Object.keys 对对象的遍历输出并不是按照属性的ASC码升序序的。

真相

根据 http://jartto.wang/2016/10/25/does-js-guarantee-object-property-order/ 的中提到的文，可以了解到：

Chrome、Opera浏览器利用 for...in 规则，会先提取所有 key 的 parseFloat 值为非负整数的属性，后根据数字顺序对属性排序首先遍历出来，然后按照对象定义的顺序遍历余下的所有属性。

这个结论可以很好地解释上面几位PK的结果。

随后，我也看了一下再看了 https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/Object/keys 中介绍的 Object.keys polyfill：

```
if (!Object.keys) {
  Object.keys = (function () {
    var hasOwnProperty = Object.prototype.hasOwnProperty,
        hasDontEnumBug = !({toString: null}).propertyIsEnumerable('toString'),
        dontEnums = [
          'toString',
          'toLocaleString',
          'valueOf',
          'hasOwnProperty',
          'isPrototypeOf',
          'propertyIsEnumerable',
          'constructor'
        ],
        dontEnumsLength = dontEnums.length;

    return function (obj) {
      if (typeof obj !== 'object' && typeof obj !== 'function' || obj === null) throw new TypeError('Object.keys called on non-object');

      var result = [];

      for (var prop in obj) {
        if (hasOwnProperty.call(obj, prop)) result.push(prop);
      }

      if (hasDontEnumBug) {
        for (var i=0; i < dontEnumsLength; i++) {
          if (hasOwnProperty.call(obj, dontEnums[i])) result.push(dontEnums[i]);
        }
      }
      return result;
    }
  })()
};
```

可以看到其实就是利用 for in 模拟的 Object.keys 的实现。

总结

- 首先， Object.keys 输出的顺序是不可靠的，对于一些要求顺序的需求，尽量避免直接使用 Objec .keys 的输出
- 如果需要规定顺序，就只能将输出的数组，再通过排序定制为需要的输出