JavaScript有关联数组吗?

2019-05-18

如果你接触过PHP，那你对关联数组一定不陌生。C或Java中数组下标都是从0开始的数值，而PHP除了数值，还可以用字符串作为数组的下标。用数值做下标的数组叫做索引数组，用字符串做下标的数组叫做关联数组，他们都是合法的数组。

```
<?php
$arr[0] = 1;         // 索引数组
$arr["a"] = "b";     // 关联数组

echo $arr[0];        // 1
echo $arr["a"];      // b
```

在JavaScript中，同样可以使用字符串来作为数组的下标：

```
let arr = []
arr[0] = 1
arr['a'] = 'b'
```

昨天，我和漂亮同事在使用JavaScript中用字符串做下标的数组时，遇到了令人困惑的问题。

缘起

在Express.js框架的路由处理中，用res.json()返回数组，下标为数值的数组可以正常返回，下标使用字符串的数组却始终返回空。这是一段最简代码，可以用来描述该过程：

```
app.get('/', (req, res) => {
  let arr = []
  arr['a'] = 'b'

  console.log(arr)  // [a: 'b']
  res.json(arr)     // []
})
```

预期返回的数组arr包含1个元素，console.log()直接在命令行打印的文本内容是`[a: 'b']`，和预期一致，然而如果通过页面请求路由，返回的内容是`[]`，这是匪夷所思的，也就是说res.json()把数组的内容吞掉了。

探寻

为了寻找问题的真实原因，我在框架的中找到res.json()方法的定义：

```
res.json = function json(obj) {
  var val = obj;
  // ...
  var body = stringify(val, replacer, spaces, escape)
  // ...
  return this.send(body);
};
```

返回内容body经过了stringify()方法处理，stringify()方法调用的是JavaScript中JSON标准库的方法JSON.stringify()：

```
function stringify (value, replacer, spaces, escape) {
  var json = replacer || spaces
    ? JSON.stringify(value, replacer, spaces)
    : JSON.stringify(value);
  // ...
}
```

那么就说明，JSON.stringify()方法的返回值，会忽略用字符串做下标的数组。为了证实这一现象，用简单的Demo测试一下：

```
let arr1 = [], arr2 = []
arr1[0] = 1
arr2['a'] = 'b'

JSON.stringify(arr1)    // "[1]"
JSON.stringify(arr2)    // "[]"
```

所以问题又来了，JavaScript标准库中的JSON.stringify()方法，为什么要忽略数组中下标为字符串的元素？是有意为之，官方不赞成使用字符串做下标，还是无奈之举，存在不可抗拒的原因无法实现？为了找到问题的根源，我试着从Chrome解析JavaScript的 V8引擎 中寻找JSON.stringify()的定义。

V8引擎是用C++写的，关于JSON.stringify()的定义应该是这一段代码：

```
// ES6 section 24.3.2 JSON.stringify.
BUILTIN(JsonStringify) {
  HandleScope scope(isolate);
  JsonStringifier stringifier(isolate);
  Handle<Object> object = args.atOrUndefined(isolate, 1);
  Handle<Object> replacer = args.atOrUndefined(isolate, 2);
  Handle<Object> indent = args.atOrUndefined(isolate, 3);
  RETURN_RESULT_OR_FAILURE(isolate,
                           stringifier.Stringify(object, replacer, indent));
}
```

可以推测出，object即JSON.stringify()处理并返回的内容，返回之前使用args.atOrUndefined()方法进行包装。这里atOrUndefined()被反复调用，传入两个参数，可以理解为，第一个参数isolate保存有完整的参数信息，第二个参数是数据的索引，结合起来便是atOrUndefined()方法要处理的完整数据。

然后看atOrUndefined()的定义，在下面的代码中，tOrUndefined()调用了at()方法，at()方法又调用了Arguments的at方法：

```
Handle<Object> atOrUndefined(Isolate* isolate, int index) {
  if (index >= length()) {
    return isolate->factory()->undefined_value();
  }
  return at<Object>(index);
}

Handle<S> at(int index) {
  DCHECK_LT(index, length());
  return Arguments::at<S>(index);
}
```

Arguments::at()方法中，指针value获取了待处理参数的内存地址，然后使用reinterpret_cast对value的值进行类型强转。

```
Handle<S> at(int index) {
  Object** value = &((*this)[index]);
  // This cast checks that the object we're accessing does indeed have the
  // expected type.
  S::cast(*value);
  return Handle<S>(reinterpret_cast<S**>(value));
}
```

到这里值就返回了，但是并没能解释为什么使用字符串做下标的数组内容会被忽略。只要是同一个数组，它的值就会保存在一段连续的地址空间中，即使reinterpret_cast处理的是指针变量，也应该无论多少都照常输出才是。

真相

最后，通过Google找到了一个关于数组使用字符串做下标的问题和答案（String index in js array），我才明白为什么字符串做下标的数组如此特殊，因为JavaScript里压根就没有关联数组！

```
let arr1 = [], arr2 = []
arr1[0] = 1
arr2['a'] = 'b'

arr1.length     // 1
arr2.length     // 0
```

给一个数组使用字符串作为下标赋值后，数组的长度不会改变，赋的值并没有作为数组元素储存到数组里。使用字符串作为下标能够正常对数组取值赋值的原因是，JavaScript将字符串作为数组的属性进行了储存。

```
let arr = []
arr['a'] = 'b'

arr.hasOwnProperty('a')   // true
```

因此，JSON.stringify()处理的是数组的内容，reinterpret_cast也只是基于指针对数组内容进行类型转换，属性什么的，当然不会有输出！

后续

1. 为什么console.log()可以将数组的属性也输出？对于要输出的内容，它是怎么定义的？
2. 为什么JavaScript中`typeof []`的值是`"object"`，也就是数组的类型是对象，但对象的属性会被处理，而数组不会？