js- 不用 extends 实现扩展原生对象

作者：bylx
原文链接：https://ld246.com/article/1682584689245
来源网站：链滴

Why not Extends?

虽然方法比较简陋，但是效果是不错的，理论性能也在我知识范围里最大优化了。至于你问我为什么用extends，我只能说

1. 我喜欢function作为构造函数

2. class兼容性有点拉

什么破原型链

最近在做在线二进制编辑器，快做完了(80%)感兴趣可以蹲一波；然后遇到了dataview和历史记录的缘，剪不断理还乱——我必须扩展dataview来让setUint8这类的函数推送历史记录来正常使用ctrl+z能。正常写法是直接在历史类里绑定dataview然后修改dataview的setUint8函数，但是有2个问题-

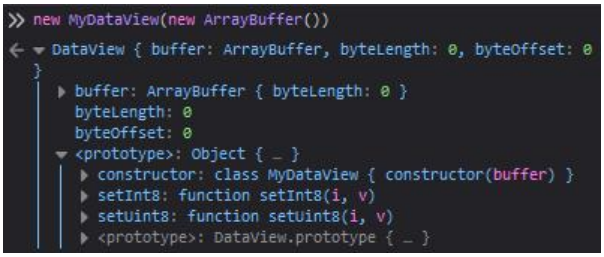
1. 当二进制文件长度变化，dataview也要随之变化，你在历史里绑定的dataview指针就丢了，没法到更改后的dataview。再更改一遍也行，但你看第二个问题

2. 绑定一个dataview就要把所有set相关的function重写一遍，全部写进了dataview的对象里，不仅制台里看着心痒而且每次绑定都要写10来个函数的话理论性能也不过关。(高标准严要求 确信)

最好的想法应该是有一个prototype代劳，把修改后的方法放在一个prototype里，然后每次datavie更新直接连接到这个prototype上就不用浪费太多内存在绑定函数上了。想一想，用extends的思路么做？无非就是

```
 class MyDataView extends DataView{
  constructor(buffer) {
    super(buffer);
  };
  setUint8(i,v) {};
  setInt8(i,v) {};
}
```

打开控制台new一个看看



看到继承链是 this -> MyDataView -> DataView

这真的能不用extends吗

不用extends的话是不是就要考虑 DataView.prototype.constructor.call了？可惜作为高贵的原生对，并不能用对象来作为这个this值，否则 TypeError: calling a builtin DataView constructor without new is forbidden

不用 call和 extends的话想要实现这样奇奇怪怪的继承链，而且还要求返回值是个 DataView (用来给 DataView 原型上方法的 this)，显然学习成本是巨大而多余的，更何况这样的需求在百度很难找

第二个，因此我把原型链改良一下：this -> MyDataView ，在this中创建一个DataView，在setUint时执行完自己的代码直接把this里的DataView传给 DataView.prototype.setUint8.call(dataview, ...arguments)。

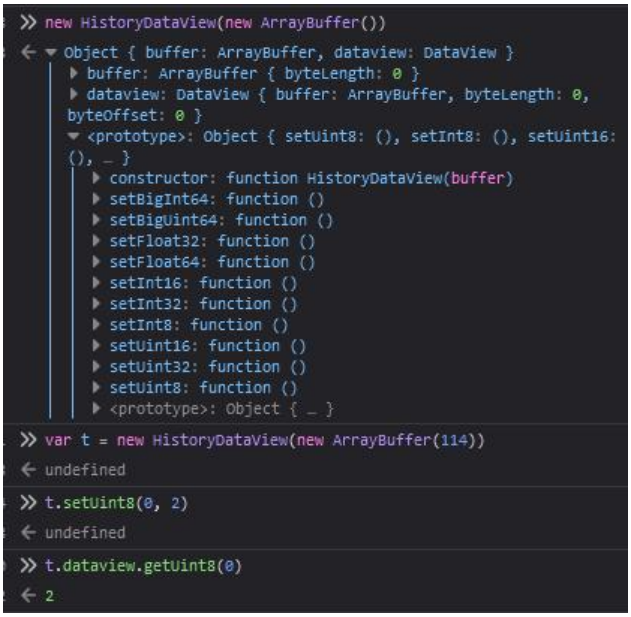
你可以理解为 DataView 的实例 new DataView(buffer)是王子，住在王宫(DataView.prototype)里可以自由进出王宫。但你只是个杂鱼百姓，想要进入王宫拿点米 DataView.prototype.getUint8吃。可以抓一个人伪装成王子 var dataview = new DataView(buffer)潜入王宫 DataView.prototype.setint8.call(dataview, ...arguments)，然后就能正常拿到你的结果。

真有这么简单我就信了

论性能而说，这些函数都绑定在 prototype上，不存在一堆 匿名函数声明乱挤内存，所以理论性能应是足够优化了(对js内存知识储量到此为止了，有错误请提出)。所以实际代码写法如下

```
function HistoryDataView(buffer, history) {

  this.buffer = buffer;
  this.history = history;
  this.dataview = new DataView(buffer);

}

["Uint8", "Int8", "Uint16", "Int16", "Uint32", "Int32", "BigInt64", "BigUint64", "Float32", "Float64
].forEach((type)=>
HistoryDataView.prototype["set"+type] = function() {

  var [i, v, l] = arguments;
  this.history.push(...); // 这里写你自己的代码
  DataView.prototype["set"+type].call(this.dataview, i, v, l);

});
```

new一个看看效果

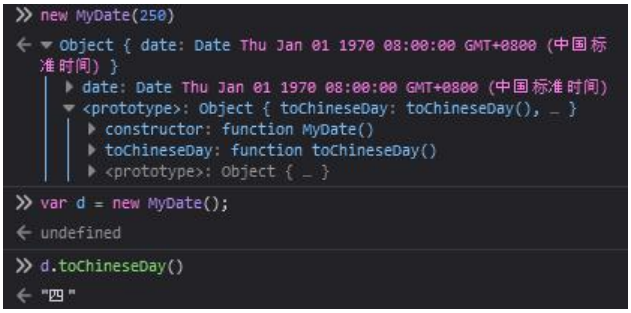


正常运行。

这么厉害，那我试试另一个

同理啊，可以试一试扩展一个自己的 Date

```
function MyDate() {
  this.date = new Date(...arguments);
}
MyDate.prototype.toChineseDay = function () {
  return ["日", "一", "二", "三", "四", "五", "六"][this.date.getDay()]
}
```



看看表，今天是不是星期四?