知乎

JS 的 new 到底是干什么的?

方应杭
四年打工，七年创业。

关注他

1133 人赞同了该文章

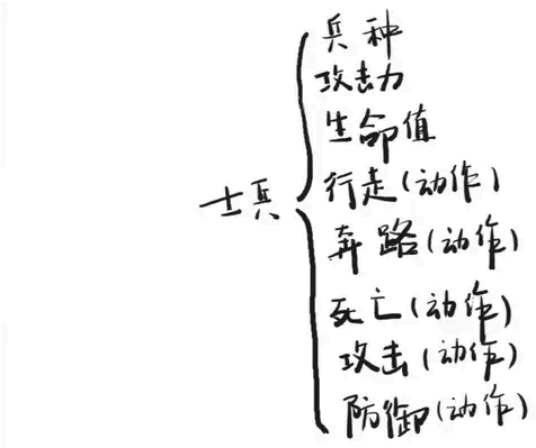
大部分讲 new 的文章会从面向对象的思路讲起，但是我始终认为，在解释一个事物的时候，不应该引入另一个更复杂的事物。

今天我从「省代码」的角度来讲 new。

想象我们在制作一个策略类战争游戏，玩家可以操作一堆士兵攻击敌方。

我们着重来研究一下这个游戏里面的「制造士兵」环节。

一个士兵的在计算机里就是一堆属性，如下图：



我们只需要这样就可以制造一个士兵：

```
var 士兵 = {
  ID: 1, // 用于区分每个士兵
  兵种:"美国大兵",
  攻击力:5,
  生命值:42,
  行走:function(){ /*走俩步的代码*/},
  奔跑:function(){ /*狂奔的代码*/  },
  死亡:function(){ /*Go die*/    },
  攻击:function(){ /*糊他熊脸*/   },
  防御:function(){ /*护脸*/       }
}

兵营.制造(士兵)
```

制造一百个士兵

如果需要制造 100 个士兵怎么办呢?

循环 100 次吧：

```
var 士兵们 = []
var 士兵
for(var i=0; i<100; i++){
  士兵 = {
    ID: i, // ID 不能重复
```

```
    生命值:42,
    行走:function(){ /*走俩步的代码*/},
    奔跑:function(){ /*狂奔的代码*/  },
    死亡:function(){ /*Go die*/    },
    攻击:function(){ /*糊他熊脸*/   },
    防御:function(){ /*护脸*/       }
  }
  士兵们.push(士兵)
}

兵营.批量制造(士兵们)
```

哎呀好简单。

质疑

上面的代码存在一个问题：浪费了很多内存。

1. 行走、奔跑、死亡、攻击、防御这五个动作对于每个士兵其实是一样的，只需要各自引用同一个函数就可以了，没必要重复创建 100 个行走、100个奔跑……
2. 这些士兵的兵种和攻击力都是一样的，没必要创建 100 次。
3. 只有 ID 和生命值需要创建 100 次，因为每个士兵有自己的 ID 和生命值。

改进

看过我们的专栏以前文章（JS 原型链）的同学肯定知道，用原型链可以解决重复创建的问题：我们先创建一个「士兵原型」，然后让「士兵」的 __proto__ 指向「士兵原型」

```
var 士兵原型 = {
  兵种:"美国大兵",
  攻击力:5,
  行走:function(){ /*走俩步的代码*/},
  奔跑:function(){ /*狂奔的代码*/  },
  死亡:function(){ /*Go die*/    },
  攻击:function(){ /*糊他熊脸*/   },
  防御:function(){ /*护脸*/       }
}
var 士兵们 = []
var 士兵
for(var i=0; i<100; i++){
  士兵 = {
    ID: i, // ID 不能重复
    生命值:42
  }

  /*实际工作中不要这样写，因为 __proto__ 不是标准属性*/
  士兵.__proto__ = 士兵原型

  士兵们.push(士兵)
}

兵营.批量制造(士兵们)
```

优雅?

有人指出创建一个士兵的代码分散在两个地方很不优雅，于是我们用一个函数把这两部分联系起来：

```
function 士兵(ID){
  var 临时对象 = {}

  临时对象.__proto__ = 士兵.原型
```

知乎

```
  return 临时对象
}

士兵.原型 = {
  兵种:"美国大兵",
  攻击力:5,
  行走:function(){ /*走俩步的代码*/},
  奔跑:function(){ /*狂奔的代码*/  },
  死亡:function(){ /*Go die*/    },
  攻击:function(){ /*糊他熊脸*/   },
  防御:function(){ /*护脸*/       }
}

// 保存为文件：士兵.js
```

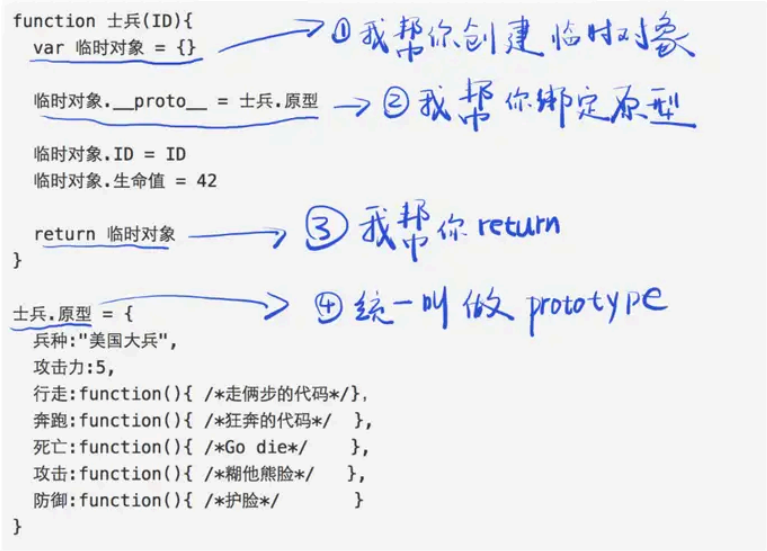
然后就可以愉快地引用「士兵」来创建士兵了：

```
var 士兵们 = []
for(var i=0; i<100; i++){
  士兵们.push(士兵(i))
}

兵营.批量制造(士兵们)
```

JS 之父的关怀

JS 之父创建了 new 关键字，可以让我们少写几行代码：



只要你在士兵前面使用 new 关键字，那么可以少做四件事情：

1. 不用创建临时对象，因为 new 会帮你做（**你使用「this」就可以访问到临时对象**）；
2. 不用绑定原型，因为 new 会帮你做（new 为了知道原型在哪，所以指定原型的名字为 prototype）；
3. 不用 return 临时对象，因为 new 会帮你做；
4. 不要给原型想名字了，因为 new 指定名字为 prototype。

这一次我们用 new 来写

```
function 士兵(ID){
  this.ID = ID
```

知乎

```
士兵.prototype = {
  兵种:"美国大兵",
  攻击力:5,
  行走:function(){ /*走俩步的代码*/},
  奔跑:function(){ /*狂奔的代码*/  },
  死亡:function(){ /*Go die*/    },
  攻击:function(){ /*糊他熊脸*/   },
  防御:function(){ /*护脸*/       }
}

// 保存为文件：士兵.js
```

然后是创建士兵（加了一个 new 关键字）：

```
var 士兵们 = []
for(var i=0; i<100; i++){
  士兵们.push(new 士兵(i))
}

兵营.批量制造(士兵们)
```

new 的作用，就是省那么几行代码。（也就是所谓的语法糖）

注意 constructor 属性

new 操作为了记录「临时对象是由哪个函数创建的」，所以预先给「士兵.prototype」加了一个 constructor 属性：

```
士兵.prototype = {
  constructor: 士兵
}
```

如果你重新对「士兵.prototype」赋值，那么这个 constructor 属性就没了，所以你应该这么写：

```
士兵.prototype.兵种 = "美国大兵"
士兵.prototype.攻击力 = 5
士兵.prototype.行走 = function(){ /*走俩步的代码*/}
士兵.prototype.奔跑 = function(){ /*狂奔的代码*/  }
士兵.prototype.死亡 = function(){ /*Go die*/    }
士兵.prototype.攻击 = function(){ /*糊他熊脸*/   }
士兵.prototype.防御 = function(){ /*护脸*/       }
```

或者你也可以自己给 constructor 重新赋值：

```
士兵.prototype = {
  constructor: 士兵,
  兵种:"美国大兵",
  攻击力:5,
  行走:function(){ /*走俩步的代码*/},
  奔跑:function(){ /*狂奔的代码*/  },
  死亡:function(){ /*Go die*/    },
  攻击:function(){ /*糊他熊脸*/   },
  防御:function(){ /*护脸*/       }
}
```

完。

想学前端？加我微信 frank_fang，加好友暗号：new

编辑于 2020-11-11 01:07

知乎

▲ 赞同 1133 ▼ 121 条评论 分享 喜欢 收藏 申请转载 ···

欢迎参与讨论

121 条评论 默认 最新

知乎用户rQd3o3
我不管，先赞一波。
2016-12-11 回复 16

冴羽
除了节省的4步之外，是不是还应该有 士兵.call(临时变量)
2017-02-24 回复 9

PeterLee
方老师，建议您直接上代码，毕竟会到知乎看您文章的基本都多少能懂点，用中文看起来挺乱的，影响阅读，个人拙见，不喜请忽略
2016-12-14 回复 9

五熊
这个中文也是可以运行的。。。
2022-10-19 回复 喜欢

Listen Hua
同
2022-09-02 回复 喜欢

jsm1003
让我想起了以前家里做月饼的时候。new就是从一个设计好的月饼模子里，pia！拍出一个月饼来的一个过程😂
2016-12-12 回复 7

XR潮汐
new做的4件事：
1，创建临时对象
2，把this绑定到该临时对象
3，把prototype绑定到该临时对象
4，return 该临时对象
2022-02-10 回复 1

海中月是天上月
有没有一种可能， 会有类似花木兰这样的女兵

2022-02-09 回复 1

无独有偶
new让我误认为js有类😂😂😂
2016-12-12 回复 喜欢

陈楠 ‣ 喵喵
JS 的 class 只是个关键字，至于类，它还是靠原型
2023-12-22 回复 喜欢

喵喵
js的确有class啊，不过底层好像依然是原型
2023-02-16 回复 喜欢

大尧
如何才能学到这样的知识?
2016-12-11 回复 12

知乎

2016-12-11 回复 6

大尧 ‣ **方应杭**
你是在怎么学到这个的呢~我也想能成为这样的人，我现在就知道怎么用，但是不知道怎么来的，每次看你的文章都感觉：原来是这样！
2016-12-11 回复 6

查看全部 12 条回复 ›

jango
这不就是高程里面的内容吗。。。
2016-12-11 回复 4

胡桃夹子
能把枯燥的教科书解释成通俗易懂平易近人的例子能为我们这些普通人节省理解的力气，这就是独一无二的能力。
2018-05-08 回复 20

方应杭 作者
是吗，我手画的图，高程可没有
2016-12-11 回复 5

小云云不小
从此我的评论只有好评和不评论-我就是那个说你写的啰嗦的人。
2016-12-12 回复 2

方应杭 作者
你觉得啰嗦那是因为你已经知道 new 的作用了。 所以你不是我的受众。 所以……是的，你可以不评论。 如果你不懂 new，看了之后还是不懂，就可以差评。
2016-12-12 回复 10

方应杭 作者 ‣ **wang z**
那我就不知道什么值钱了，我是面向赞数写文章。
2016-12-12 回复 4

查看全部 6 条回复 ›

点击查看全部评论 ›

欢迎参与讨论

推荐阅读

skynet的xmake构建脚本

skynet是一个轻量级的游戏服务端框架，代码非常简洁，连构建也继承了这种简洁性，只要敲下make <plat>即可完成相应平台的编译。那即然这么方便，为啥还要搞个xmake脚本呢？其实主要原...

co li... 发表于游戏服务器...

xmake 源码架构剖析

ruki

C/C++ 构建系统，我用 xmake

ruki 发表于TBOOX...

Xmake v2.6.6 发布，分布译和缓存支持

ruki 发表于TBO