开篇词 | 如何解决语言问题?

2019-11-11 周爱民

《JavaScript核心原理解析》 课程介绍 >

讲述：周爱民

时长 10:26 大小 9.56M

你好，我是周爱民，和你一样，我喜欢 JavaScript。

我是《JavaScript 语言精髓与编程实践》这本书的作者，这个书名正好也刻画了我追随 JavaScript 的轨迹：在过去的二十年中，我一面研究它的语言精髓，一面做编程实践。

曾经在很长的时间里面，我的脑海中时常会有一个闪念：如何解决语言问题？这也伴随着强烈的自我否定与质疑，因为它的潜台词是：我还没有搞定语言问题。

问对问题

在那之前，我是从 DBASE 这个数据库入手，从类似 SQL 的语言开始学习的。第一门正式学习的语言是汇编，然后是 Basic 和 Pascal。后来在商用产品开发的环境中，我选择了 Delphi 这门语言。这一段语言学习的经历，直到 2003 年前后戛然而止，那时我写完了我的第一本书，名字就叫《Delphi 源代码分析》。

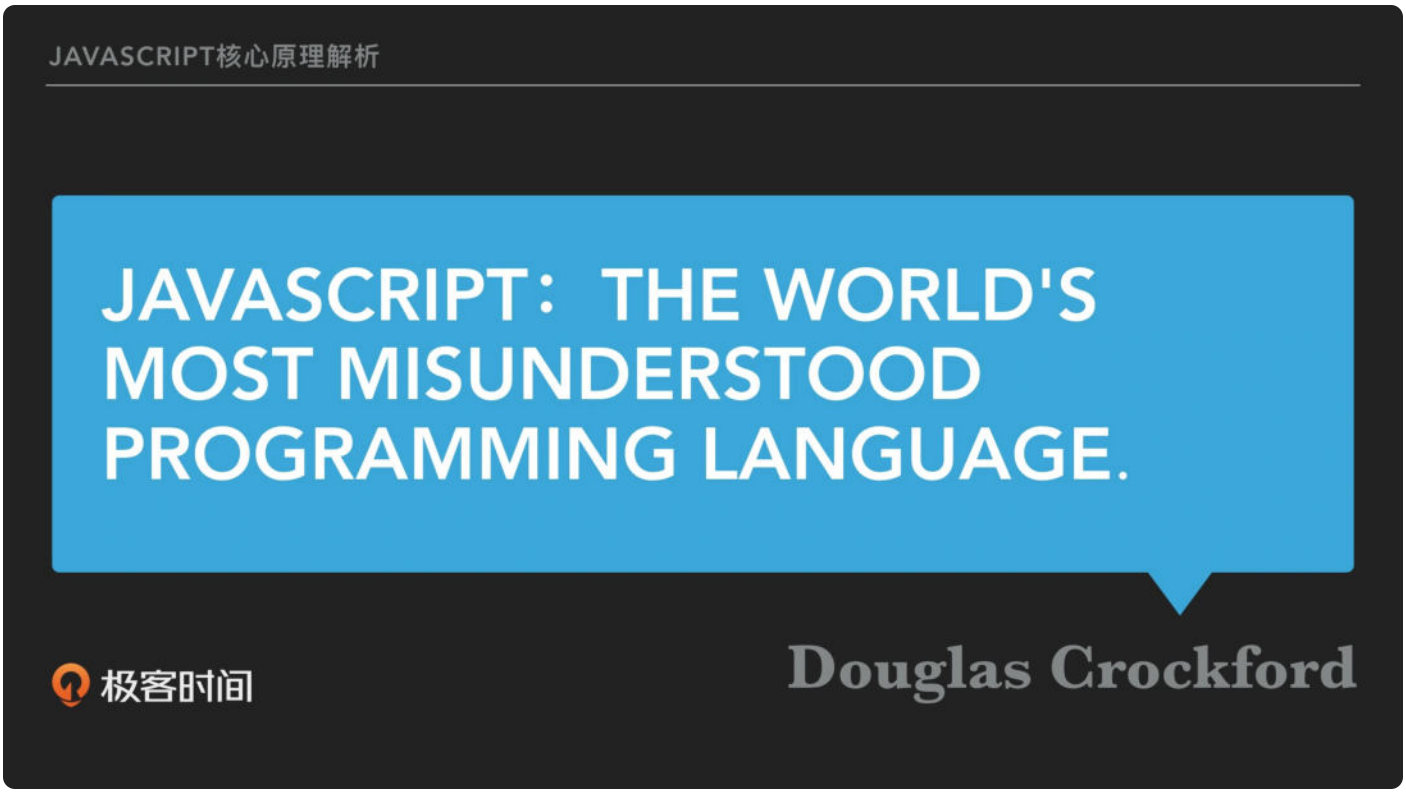
这些是我既有的语言知识，一定程度上来说，它限制了我对 JavaScript 的进一步学习，既成习惯的思维方式和知识体系盘根错节，渗透极深。

我是从 1998 年左右就开始接触 JavaScript 这门语言的，然而直到五六年之后，我仍然在使用 Delphi 中的面向对象概念与理论来理解与分析 JavaScript 的对象模型。这也是我早期做 WEUI 这个项目时的困扰：我一面努力改造着 JavaScript 这门语言，一面又被它所左右。

那个时代，有太多人做着与我相似的事情：要上线一个大的前端产品，就先写一个框架或库，将传统语言中的经验复制过来。那些是我们的既得财富，闪烁着记忆的光芒、知识的火花，是自我价值和薪资的体现，所以它们是不可抛弃的、不可否认的、不可亵渎的。

在类似于此的、固化的思维里面，我们勤劳地、不知疲倦地写着新代码，而究其原因，只是我们不愿意丢弃那些旧代码而已。

如此负重前行，以至于让我不得不怀疑，自己是否还有能力掌握这门“世界上最被误解的语言”。这是 Douglas Crockford（就是创建了 JSON 格式那位大牛）在 2001 年说过的话。



我真正下定决心抛弃所有，来重新理解这门语言的时候，是在我写出《Delphi 源代码分析》之后不久。这个时候，Borland 破产了，Delphi 被卖掉了，而我也做了架构师。此时，那些既有的经验，以及对语言孰优孰劣的固执己见都变得可有可无了。这时，对于我一直以来的困惑，我才真正地问对了第一个问题：

JavaScript 到底是一门什么样的语言？

语言

我常常说，问对了问题，也就有了“解”。

在 JavaScript 诞生的时候，主流的应用开发语言大多是静态的，以及单一语言类型的。当时面向对象编程大行其道，众多语言都纷纷以“**我最 OOP**”为宣传噱头，以及将它想像成语言发展的必然方向。随着 Java/JVM 的成熟，使用中间指令集 + 虚拟机来运行的语言环境也变得流行起来，因此一个虚拟机上跑很多种语言也就成了常态。但即使如此，具体到每一种语言，其主要特性还是单一的，并且通常以保持“**语言特性的纯净**”为己任。

JavaScript 却是一个异类，它最开始是一门“简单”的小语言，没有丰富的语言特性，也没有大一统的野心，更没有“包打天下”的虚拟机引擎。为了维护这种“小而简洁”，当然，另一部分原因也在于它的创始者太过匆忙和随意——它只实现了一些基础的语言特性，而没有从根本上“陈述”自己的设计原则与理念。

这门语言非常摇摆，在面向对象火的时候，它说“我是 OOP 的语言”；在函数式语言呼声渐高的时候，它又说“我是函数式的”。另外，你知道的，它天生还是一门动态的语言。不过，它也还包括一些静态的、结构化的语言成分。

支持这门语言挣扎求生、一路行来的，正是这门语言最初的、最精彩的设计：**它是一门多范型语言，或者，也称为混合范型语言**。JavaScript 的简单来自于此，复杂也来自于此；生存能力来自于此，抨击诟病也来自于此。

的确，如果我不抛开 Delphi 语言给我留下的历史包袱，我还是能从 JavaScript 中看到我熟悉的、引以为傲的经验闪光，并让这些东西迅速激活我对语言的兴趣和掌控感。然而只需要稍稍多一点点时间，混合语言中的其它组分就会变成我的困扰，变成这门语言给我带来的种种陷阱，变成我近乎绝望的自我怀疑。

而其实问题的求解也很简单：不要试图去纯化这门语言。

语言的特性

所以在这个专栏里面，我就想与你讲一讲我对 JavaScript 各种语言特性的理解，还有展示将这些语言特性和语言范型融合如一的具体挑战与折衷。

JavaScript 主要包括 5 个方面的语言特性：结构化编程、面向对象编程、动态语言、函数式语言和并行语言。因此在这个专栏中，我将以“语言”为核心，主要讨论语言设计，结构化和面向对象特性，以及部分的动态语言特性。还有一些其它部分的特性，我将会在以后的专栏中再给你讲。

在讲述的内容方面，尽管每一讲都是一个独立的标题，但总体来说，这些内容也是循序渐进的，因此你最好不要落下课程。并且如果有时间、有机会的话，还是对前面的内容做一些分析和巩固为好。

另外，你可能已经注意到了：每一讲的标题都是一行代码。尽管这些代码绝大多数都是有意义的、可以使用的，但是我并不是从实用性出发来写出这些代码的，因此它们不见得能很好地使用在你的项目中。我尽量使每一讲的标题在表达多种语言特性的同时，指向一个主要的、核心的内容讲述方向。

事实上，如果你看到那样的一行标题后，能猜出它涉及到哪些混合语言特性，或者是由哪些特性共同作用以导致这个代码的形式风格或可能的结果，那么你也就相当于复盘了你的知识储备，这有助于你建立自己的知识体系。

所以，你大可以将这样的标题当作一把念珠，没事的时候，盘一盘。：）

体系性

说到体系性，其实这才是我这个课程最关注的地方。

我希望综合这些代码的特殊性，代码所涉问题的领域，代码的逐步分解解析，以及辨析与该代码相似的或同类的问题，一方面发掘它们潜在的应用，另一方面，则在于帮助你构建一个语言知识结构。这样的语言知识结构，能让你看到曾经摸过的那些项目，写过的那些代码，填过的那些巨坑的影子，并且发掘暗影背后涌动的语言原力，找到属于自己的、可规划的语言学习体系。

你不需要精通所有的语言，但如果你了解那些语言类型的核心的、本质的差异，建立起自己的对语言的认识观和辨析力，那么当你接触到一门新的语言时，便可以在极快的时间内将它纳入自己的语言知识结构，快速地映射到那些历史项目和研发经验中。你可以通过想象，将新的语言在自己的经验中“回放一遍”，这相当于用新语言重写了一遍那些代码，也相当于将你自己的历史经验全部消化在这个新语言之中。

这样的语言学习和感悟方法，收效是极快的。

我曾经在豌豆荚的工作中，完成过一个用 Lua 来实现的、支持大规模并发的服务端项目，那是一个金融级的风控产品。在我对 Lua 了解几乎为零的情况下，出于对“编程语言核心原理”的了解，通过上述知识映射的方法，我在零学习的情况就开始了编码工作。除了必要的查手册之外，切换语言的时间成本几乎可以忽略不计。

当然，听到这里的时候，你可能会说“Lua 和 JavaScript 的相似性太高”，又或者说“Lua 太简单了”。但是事实上，我之前在学习 Erlang 的时候也是如此，以及后来在学习 Scala 的时候仍然是如此。当你真正地“解决了语言问题”时，“新语言”真的对你来说完全不能称其为问题。

所谓的语言特性，其实是对语言的核心抽象概念的语法表现。所以，所谓的“学习新语言”，只不过是在玩“变换代码风格”的游戏，而已。

一旦你建立了你的体系性，那么相当于你创建了游戏规则，你就成了“编程游戏”中的上帝。

你将会有一种切实的、万物如一的操控感。

学这门课

所以，这门课的内容大概率不会用在你的一个线上项目中，也不会提高你写代码的速度。但**它绝对能让你提升对代码的洞察力**，让你可以在纷繁的代码中快速找到它在性能、组织、逻辑等问题的关键，也可以在语言层面给出合理的解释。

当然，就一个具体问题的具体解，这一切还是不够的，因为语言是实现业务的工具，而不是业务本身。所以，在面试中不仅仅会考察你的语言能力，也会考察你对曾经项目的经验与感受。

无论如何，我希望你能找到自己对语言的认识，无论是不是通过这门课程，“构造认识”对你自己而言都是极致重要的事情。如果在这其中，我的课程能对你有所帮助，哪怕仅仅是一点点启发，我想，这都是我非常乐意看到的结果了。

最后，多谢你来看我的专栏。关于 JavaScript，你的理解是怎样的呢？你又有什么样的期待？欢迎留言。

下一篇　01 | delete 0：JavaScript中到底有什么是可以销毁的

精选留言 (49)

写留言

crown
2019-11-12

老师好，我工作主要用node. js开发游戏服务器和web服务器的，按照我对操作系统的理解，单线程容易写对逻辑，非阻塞io如文件操作等+ epoll可以实现异步io，开多进程可以利用多核心。 可是大家都说node是单线程做游戏服务器不行，但是游戏逻辑不太好用多线程表达吧。。。 说node不适合写游戏服务器，主要说是js运行效率的缺陷?

作者回复: 在涉及纯计算效率的问题上，js确实是要差一些的，要不然也不会专门再弄出一个BigInt类型出来，以及还有所谓的TypedArray。但是，如果仅仅是说能不能做游戏服务器，开发游戏应用代码，那么坦率的说：1、没问题，2、效率还错，3、开发挺快的；4、网络实现也很简洁……但是如果游戏需要处理传输层的编解码，等等与计算效率相关的情况，那么（前面已经说过了）就得换方案了。

你提到多线程问题，对于最新的Node.js来说，原生多线程也是内置的了，但并不比传统开发工具下的简单。——所以除了语言或工具的习惯性之外，它也并没有什么优势。另外，在线程操作这个层级上，大家提供的都是差不多的核心操作或API，所以使用复杂度，面临的问题集之类的也差不多。只不过，相对于其它语言或应用环境，Node.js缺少在多线程基础上的开发库，这方面基本为0。

最后，还是从纯语言的角度上来回答你的问题。事实上决定它合不合适写游戏的，根本之处（从语言的角度上来讲）与你所提的绝大多数特性都无关。如果游戏规模大一些，那么就不适合直接用JavaScript来开发了。在任何的、包括游戏开发在内的大型的应用开发中，JavaScript最大的缺陷其实是缺乏静态类型声明与检查。在动态类型体系下开发大型程序几乎是一种灾难，即使你再能控制类型的使用、团队的编码习惯等等，都很难避免。

因此如果你非得在这种场景下使用JavaScript，我非常建议你考虑TypeScript，哪怕是Flow也行。当然，最终运行的执行环境仍然可以是Node.js，这就是个效率问题了，而与语言无关。

41

烤迪
2019-11-15

老师可以推荐一些有利于洞悉语言本质的书吗，可以结合专栏阅读或者后面继续学习的书

作者回复: 裂墙推荐。^^.
https://github.com/aimingoo/my-ebooks#程序原本

老实讲，比这里用js来讲的内容要易读得太多了。这里讲js吧，我还得注意“js/ecma到底如何实现”，或者这个特性那个特性什么的。而《程序原本》里只讲本质就好了。^^.

共 4 条评论>

27

小伟
2019-11-11

js是我最早接触的语言，也是一直敬畏的语言，你以为掌握了，但在debug的时候，error会给你致命一击

15

余文郁
2019-11-12

```
function Foo() {
  getName = function() {
    console.log(1);
  }
  return this;
}
Foo.getName = function() {
  console.log(2);
}
Foo.prototype.getName = function() {
  console.log(3);
}
var getName = function() {
  console.log(4);
}
function getName() {
  console.log(5);
}
Foo.getName();
getName();
```

Foo().getName();

getName();

new Foo.getName();

new Foo().getName();

new new Foo().getName();
老师，我做了一道网上的题目，发现做不出来，运行后，发现在node环境会报错，浏览器环境会依次输出2 4 1 1 2 3 3，老师可以帮我解释一下吗，确实有点懵了

作者回复: 按浏览器上的执行效果理解就对了。

NodeJS加载的文件是运行在一个模块中，因此function Foo()的代码中的"return this;"并不能得到global，从而导致“Foo().getName();”这一行不能得到global.getName出错。

共 3 条评论> 6

lunar
2019-11-24

小岳岳不仅会说相声，js还这么好? 😱😱

6

余文郁
2019-11-20

老师，今天突然想到一个问题，就是为什么js中有高阶函数这个东西，可以把函数当参数，还可以返回一个函数，其他语言里没有，这跟js诞生有什么联系吗

作者回复: 函数式语言特性并不是js独有的，它是一种被广泛认可的语言特性。但在函数式特性方面，Eich没有太强调他在早期语言设计中受到何种影响。

现在的tc39中的成员们，从整体趋势上来看，是更认可js中的函数式特性，而否定其中的oop特性的。
所以，有时候感觉这门语言（的特性）已经开始变得怪怪的了。:(

共 2 条评论> 5

鸡蛋火腿酥饼
2019-11-14

老师您好,我现在正在一家公司实习,使用JS为公司开发功能需求的外部库,但是总觉得自己的代码风格和效率差,不知从哪学起,也不知学您这个适合吗?

作者回复: 如果是这个目的，那么不太适合。因为这个课程的内容，讲工程应用的较少，对代码风格也缺乏讨论。

这个课程更多的是构建语言的学习与认知的体系，而不是提供应用开发的技能，二者目的不太相同。

4

张三

2019-11-14

老早就看了爱民老师的绿皮书。两个字“牛B”。

共 2 条评论>

4

ssala

2019-11-12

接触的语言中，js是唯一一门让我头大的语言，打算跟随老师一起学习。

共 2 条评论>

4

温晓东

2019-11-12

爱民开课 支持 希望看见更多的课

作者回复: 谢谢支持✊

4

ericluo

2020-05-18

周老师好，您能具体分享一下用LUA做的金融级的风控产品的背景吗?
如为什么当时技术选型使用了lua脚本语言？当时的解决方案还有那些，然后最后为什么选择了lua，谢谢

作者回复: 关于选择lua，其实主要是没有太好的选项。当时的服务器环境是nginx的，如果想在server端做一些重量级的事情，那就只有nginx_lua好用，当时的nginx_javascript官方支持已经出来了，我考察了一下，很不成熟。所以很大程度上，选lua只是因为如果需要“运行在nginx中”，那么只有他跟nginx配合得很好。

有些风控方案是将风控系统运行在外部的，这种情况下nginx_lua就不是必须的，也不是最优的。而我

当时需要做到实时的风控，因为金融级的必须在最短的时间实现反向控制。所以最可行的选择，就是在nginx内加入监控点、ai决策和反向控制逻辑。

另外，关于这个产品你可以看一下ngx_cc这个项目，在这里：
> https://github.com/aimingoo/ngx_cc

在它的wiki里有一些简介和架构参考：
> https://github.com/aimingoo/ngx_cc/wiki/简介

共 2 条评论>

4

翅膀
2019-11-12

第n+1次，学js. 这次要坚持住

作者回复: 加油！

3

翰弟
2019-11-14

早闻其名，以为老师曾已经输出过那么多，以为这次课可能没那么多真知，听了开篇词我先保证学完这个课。
另外发现技术科程的订阅量和得到之类的课不同比。我决定让女票也看这开篇词，接收“新”东西。
我是一个好不纯粹的程序员

1

Jack Q
2019-11-11

掌控代码，而不是被代码掌控。

作者回复: ^^.

完全赞同楼上观点

3

一步
2019-11-11

JS这门语言成也灵活，败也灵活

熟练的怎么写怎么舒服，不熟悉的人感觉没有规范太灵活了容易出错

希望通过这个专栏形成语言的体系

2

潇潇雨歇
2019-11-11

上个月就在期待了，看了课程预告那会

2

潇潇雨歇
2019-11-11

一直期待着这门课程，终于来了；我觉得js是一门坑比较多的语言，因为当初作者设计编写的时候比较匆忙以及领导意思，但是毫不影响它的优秀，我们应该尽量去其糟粕，搞懂核心和特性。再也不怕了。然后就是ES6的到来，js变得越发强大了，我期待他能更加优秀，越来越强大。

2

勿闻轩外香
2019-11-11

今天立flag，没事儿的时候盘一盘：）

2

kkxue 魔
2021-05-10

爱民老师，请指点下如何阅读《编程原本》

作者回复: 从头到尾，读到读不懂为止。

这本书是由浅入深的，由抽象到具体讲的，所以越早的部分跟你现有的知识重合度越高（越易懂）。

但是因为越早的越抽象，所以有“似是而非（看起来像是，细读下觉得说得不同）”之感，这种情况下要多思考，但不见得是你理解得不正确。

如果往后，有读来觉得“知而不解（就是看起来知识点都知道，但是不懂作者表达的意思）”，那就是你现在知识体系中理解得有误的部分了，忘掉已知的，跟随文中的思维逻辑重新理解一遍。

最后的部分（尤其是“系统”相关的部分），那是架构师的入门之径，如果读不懂也不要紧，看下，记住，要用的时候自然就懂了。

这本书的后续是《我的架构思想：基本模型、理论与原则》，那是按架构师的要求标准来写的，读法就更不一样了。

共 2 条评论 >

2

undefined

2021-04-23

– [LINK](https://time.geekbang.org/column/article/163480)
– 这个专栏我听了和看了多遍，同样的内容，每次学习的感受却不尽相同，总会发现一些新的知识点或是新的理解。
– 不同的人，对于不同学习形式的接受能力也是不同的。有些人喜欢阅读书籍，有些人则更倾向于视频教程。对我自己而言，我并没有特别大的偏好。
– 针对这个专栏，我通常是先粗略的听几遍，了解大概内容，然后再细读原文。再听，再阅读，周而复始。阅读时速度更快，会下意识的跳过部分解释，偶尔会错过重要的知识点。而听的时候，往往容易发现自己忽略的部分，甚至觉得自己读了假的文章。
– 对于刚接触这个专栏的同学，我个人建议先去看看加餐部分的《捡豆吃豆的学问》两篇文章，掌握学习思路后再来学习。
– 按照《捡》里提及的学习方法，首先应当自己设问。针对开篇词，我列了如下问题：
– **自己提问题**
  – 周爱民是谁
  – 什么是语言问题
  – JavaScript有哪些语言问题
  – 怎样学习JavaScript
  – 这个专栏应该怎样学习
  – 这个专栏需要什么样的基础
  – 怎样发现问题
  – 怎样发现自己的问题
– **求解答案**
  – 对于上述问题的求解因人而异，而我也没有找到所有的解答。
  – 解答的过程，应该贯穿整个学习过程，很难一次搞定。
  – 尝试用文字总结，而不只是浅尝辄止。复述和重新描述，本身也很考验自己的理解和表达。
– **学习总结**
  – 应当尽早建立自己的知识体系，宜早不宜晚。
  – 学习遇到困难，应该是件高兴的事儿。因为一旦解决了这些问题，就能获得可感知的进步和成就感。当然，在解决的过程中，倘若遇到始终无法搞定的难题，不妨直接跳过，过段时间再来尝试。如果有高人指导的话，寻求帮助也未尝不可。
  – 不妨时不时读点 Specification，从标准的角度理解 JavaScript 的实现。
– **相关链接**
  – [加餐 | 捡豆吃豆的学问（上）：这门课讲的是什么？](https://time.geekbang.org/column/article/171116)

  - [加餐 | 捡豆吃豆的学问（下）：这门课该怎么学？](https://time.geekbang.org/column/article/171125)
- **时刻准备推翻再来**
  - 为什么？
    - 相关的技术会更新，已知内容可能会失效。
    - 同样的问题，随着实践的增加，可能会产生新的感悟。
    - 存在着错误的理解，当时的自己并未发觉。

作者回复: 👍👍+5

共 6 条评论>

2