

加餐 | 捡豆吃豆的学问（下）：这门课该怎么学？

2019-11-27 周爱民

JavaScript核心原理解析

[进入课程 >](#)



讲述：周爱民

时长 17:19 大小 15.87M



你好，我是周爱民，在上一讲中讨论了这门课程所学的内容“到底是什么”。接下来，我们再来看看“怎么学这门课程”。

教的方法

我先来说说这个课的教法。有没有简单、明晰的授课方法呢？有的，你在极客时间上也好，学校的课程里也好，常见的一个教法套路便是：

开篇立个题，把问题抛出来，说我们今天要讲什么什么，关键点有一二三四；

接下来就讲这一二三四，分析也好，解说也罢，趣谈也可，总之让你听得开心有味；

最后收纳主题，我们讲了一二三四，你看看你听懂没有。

听不听得懂？能！你认真听下来，只要老师不差，绝对能懂，如果正好是你没听过的内容，还感觉耳目一开，受用无穷。

但是你仔细想想，你至多知道了老师所讲的，还能知道别的什么吗？几乎不能。清楚明白、无有疏漏，但也毫无差别，你听到的跟别人听到的，你学到的跟别人学到的，完全一样。

所以，这也就是“一般的”知识。

如同我刚才说过的：你学的跟人家一样，听到的跟人家一样，知道的跟人家一样，充其量学了个跟老师讲的一模一样的，可不就是“水平一般”么？

核心原理可不可以这样讲？你可不可以这样学？答案是，其实也是可以的。我如果是把这个课程当成一门功法，——二二地讲给你听，你全然听去了，一字不落下，那这个核心原理课也可以讲得如同流水清风，让你很是舒坦。

但是你可记得，我在这门课的开篇词里说到的，**学这门课的目的是什么呢？**我说过，我希望**你能够构建自己的“语言学习体系”**。体系性，才是所有学习中最难得的。如果你有自己学习的体系，甚至是构建体系的体系，那么学这门课又算什么难事呢？这门课真的在讲什么高深的佛法，玄妙的奥义么？都不是的，它是在讲“另一个体系”下的东西！

准确地说，这门课程的讲法，与你过去二三十年来的学习方法是两个不同的体系；这门课程的内容，与你从业经历中所熟悉的语言，也分属于不同的知识体系！既然如此，你怎么可能指望用你现在的法子，在你的体系中去理解这些知识，又或者为这些知识构建“自己的语言学习体系”呢？

所以这门课程，一开始的讲法就大不相同。

这门课程的“标题”是一行代码，它通常很奇怪，有可能很有用，也有可能根本就不能用，它本身或许就难解，就是一个“问题”。然而，你需要知道，这个“标题”，或者这个标题中的“问题”，其实一点儿也不重要，我讲课不去奔着这个问题去，你学习中也不必奔着这个问题来。“求解这个问题”根本就是一件没有什么意义的事情。

所以在从一开始听课，一直听到现在的同学中，还有一些是困于 **第 1 讲** 的“delete 0”这行代码的，希望明白这行代码在讲什么、有什么用的同学，可以暂时地收收你的思

想，因为——解决这个问题，其实没有什么大用。

既然“主题没什么用”，那么我怎么讲呢？其实我每一讲的开始，就无非是拿这个标题做个引子，然后无所谓背景、历史、相关的知识点，以及各种各样的问题纷纷地抛出来，貌似东讲讲、西讲讲，一直都绕开了这个“标题中的代码”在走。

事实上，我把这整个的过程称为“撒豆子”。

怎么“撒”呢？整个课程大概 $2/3$ 甚至 $4/5$ 的内容，就是一大把豆子！一股脑儿地撒出来，没什么章法，也没什么技巧，也没有什么道理。只有一个原则，这些豆子，都是围绕“标题”的这个话题来的，它们或是互有相关性，或者是彼此有相似性，或者……等等等等。

总之，简单地说：它们是“同一个系统”下的东西。

这是我组织每一节课程的基本原则，这个原则就是：在标题之下，东拉西扯，直到一地豆子，四处乱滚。

最后我告诉你，学习这门课程的终极秘密：

把这些碎纸片捡起来，捡起来的，就是你的体系。

学的方法

所以，这门课的听法也就不同。

你非要去盯着每节课的标题，把它弄得一清二楚，知道它怎么来的，怎么解释，以及怎么去应用到项目中，老实说，也无不可，也会有所得。但终究是“捡了芝麻丢了西瓜”。我既然说了，这“大西瓜”就是这一地的豆子，关键是在于你怎么捡，而不是在于我怎么讲。

所以，我再来讲讲这个“捡豆子”的方法。

1. 设问，列问题

我可能在讲课中会“问”一些问题，但多数情况下，那是为了讲课的上下文连贯，那些问题本身并没有太明确的指向性。而且，即使是“有指向性”，又能如何呢？你求解了，也不过是多解了一个问题，于你无益。

真正有用的，是你自己学会“提问题”。

找一张纸，列一下这个标题给你带来的问题；

列一下在这个主题下面你不知道的，或者你想知道的问题；

列一下听课过程中发现的不解的、难解的问题；

列一下你的理解跟我所讲的内容之间，那些貌似“不可调和”的概念问题。

这些仍然不够重要。更重要的设问是：

为什么会有这些问题？

这些问题指向哪个“黑暗未知的方向”？（这个方向是你的知识盲点）

老师为什么要撒这些豆子？（这些豆子有内在的相关性，而这就是我撒他们的目的）

为什么会存在跟既有知识的矛盾？

为什么在 JavaScript 语言的层面“看不到这些问题”？

为什么.....要问上面这些问题？

.....

总之，**带着问题来学习，学会从你的问题中求解**。这个过程，就已经与你之前的学习方法不同了。

是你接受“我所讲的知识”好呢？还是你“找到自己的答案”好呢？

2. 求解，在知识域中找答案

既然我在每一个大段落中划了一个知识域，那么上面这些问题也就应该在这个知识域里去求解。

比如说你有人生、事业、理想的困惑了，那么你该去找知心小姐姐，非得在这么二十讲的课程中去寻答案，你肯定是找不到的。所以，上面你可以尽量宽泛地设问，到了这个求解的时候，却应该把它限定到我们讲的这个问题域里面来。这二十讲一共有四个大主题，每个大主题是一个领域。所以你得想想，你的问题可以放在哪个领域里，为什么这么放，为什么是这个领域，为什么不在其他的领域范围内。

这件事跟主题有什么关系？

这个东西是哪方面跟其他东西“有关系”？

怎么表达这种关系？

如何把它们放在同一个体系下（逻辑下或者抽象概念下）来解释？

.....

总之，多问几个为什么。

求解、答案都可以是错的，没关系，先做着，直到你能得到一个“貌似可能的解”。

3. 推翻，找到反例，精化抽象

有了“貌似可能的解”只是个开头，如果你止步于此，那之前的努力就全部白费了。这跟“一般的学法”并没有什么不同，甚至还远远不如别的老师的教法，直接给你来个“三段式”的立题求解。真正对整个学起提到提升效果的，正是这第三步的“推翻”。

问题是你提出的，答案是你找到的，而推翻也由你来行使。

正是因为你提出问题，所以你知道“源起”；正是因为你找的答案，所以你知道“经由”。你知道一件事情的源起与经由，那么要找到这件事情的关键处，其实只需要看看那些“自相矛盾”的地方，就好了。你找到你的逻辑的、过程的、结果的任何一处反例，进而重新按上述过程来思考，重新找到“貌似可能的（第二个）解”。

如此往复，最终你就看了一些事物最初的，以及最终的面貌。

有了这个面貌，你为它命个名字，抽出个概念，于是就得到了一个“抽象”。有了抽象概念，你就可以在概念的层面上描述事物，以及进行事物的推演。而这，就是架构的基本功。

有了体系性，有了概念抽象，有了推演过程，你做的，就是体系架构的工作，而不是“写代码”。代码是你架构的表现方式，仅此而已。

我想这个过程，以及这个过程的可能的结果已经超出了多数同学的“需要”。是的，暂时的，你并不需要变成“架构师”，我这门课也并不是要教你“做一个用 JavaScript 的架构师”。

最佳实践

但是，正因为这个最后“收官”的过程比较抽象、比较虚。所以，我给你在 [第 1 讲](#) 的时候就留了个伏笔，你回顾一下，我在第 1 讲的结束的时候，提过一个问题：

delete x 中，如果 x 根本不存在，会发生什么？

这个问题在“潇潇雨歇”同学的答案后面（他的答案是正确的）。在他的答案里面，我又提了两个潜在的问题。其一是：

在（如果 x 根本不存在，delete x 什么也不做，返回 true）的这种情况下，x 是什么呢？它显然是语法可以识别的东西，但如果这样，在语法上它是什么，且在执行环境中它又是什么？

这个问题其实问得很深，也正是我们这里说的：如果你找到了“貌似可能的解”，那么就进一步地找一下反例，进一步地“精化抽象”。

为什么呢？

其实啊，我们得问一个很深层的、有些哲学性的问题：不知，是不是“知”的一种？

对于 JavaScript 来说，如果一个标识符 x “根本不存在”，那么就是真正的“不知道它存在”吗？不是的，JavaScript 必须知道——“这里一个有未知的标识符”。在 JavaScript 引擎来说，我不能确定它是什么，我的整个引擎中都找不到它，但是我必须把它“标识”出来，只有把它标识出来，我才能处理它！

所以，在语法概念上，**词法记号** (Tokens) 是比**标记** (Identifier) 更底层的抽象概念——也就是更“精化”的抽象。

但在 JavaScript 中，不需要理解所谓“词法记号”，因为它不需要在这种引擎层面的“对代码文本的理解”。而在引擎层面，是将代码文本解析成词法记号序列的：它认为，所有这样的序列——也就是一串字符，要么能解释成标识符，要么就是一个不能识别的序列。

当“不能识别的序列”出现的时候，就是语法解析期错误，简单地说，就是代码错了。接下来，当词法记号是有效的标记时，它可能是能识别的、环境中有的，也就是说它是能被引擎从环境中发现（Resolve）到的引用，因此它就称为“可发现引用

（ResolvableReference）”，反之——例如上面提到的“未声明的 x”，就称为“不可发现的引用（IsUnresolvableReference）”。

注意，这些概念不是我生造的，你在读 ECMAScript 规范时就会看到这些概念名词。只是 ECMAScript 并不解释这些概念的由来，以及它们之间的抽象关系。

所以，引擎必须能识别“不能识别的标识符”。能识别才能处理，即使这个处理“仅仅是”抛出一个异常。

你想想，要是不能识别、不能抛出异常，那么这个引擎就该出现完全未知的逻辑了，这种情况下，引擎的更外层，例如宿主程序，又例如操作系统就会无法处理了，就会中止进程。引擎要么抛出一个异常，然后退出程序；要么操作系统直接将引擎杀死，连异常也没有。

我们都是有经验的程序员，上面哪种处理更好，是一目了然的事情。而上溯整个处理过程，就在于在“精化抽象”的过程中，有没有处理“不可发现的引用”，又或者说，“未发现”是不是被当成了一个需要处理的抽象概念。

少了一个抽象概念，少了一个处理逻辑，你的程序就“莫名其妙”地退出了。如果这是一个框架，或者这是一个库，一个平台系统，这个抽象概念一少掉，那么就没有人会去使用它了。因为，你知道的，系统中怕的不是出错，而是，出了错却不知道。

“知未知”，就是这个概念系统中最顶层的抽象了。

这是一个在“概念完整性”方面的实践。

对于一个体系来说，概念完整性是很重要的，如果缺乏关键概念，那么这个体系构建就会出现漏洞。习惯性上，人们用“概念对称性”来解决这个完整性的问题，例如“能发

现的 vs. 不能发现的”，这两个概念在抽象层面上，就是指“所有的”；又例如，索引数组对应连续存储，而关联数组对应非连续存储，所以“连续的 vs. 非连续的”，就意味着“数组能处理所有存储（的数据）”。

别担心，还有

到这里，可能就有同学说了，这个讲课的方法是很新颖，学习的方法看起来也可行，但是我就是这么做的呀，问题我想不到“有效的解”啊。

对啊，如果你一次两次就能想到有效的解了，一遍两遍就学成收工了，那也只能说明这个东西还是“一般的东西”，这个方法也就是“一般的方法”，而照着这个路子做下去，你也就还是个“一般的你”。

所以，不要担心，你没学明白也正常，上面的做法找不到“有效的解”也正常，这门课听到现在，以及后面要听的内容，都无非是给你一个“使用这种学习方法”的训练营，你在这个过程中，多练多试，多出错多反思，就成了。

学习要“知味”，你一旦从这个过程中得到了收获，你就如同食髓，乐此不疲了。所以，**不要气馁，放松心态，坚持就好了。**

并且呢，这门课程后续还为大家准备了更“丰盛”的加餐。按照编辑们为你制定好的学习计划，我还会在第 10 讲之后，给大家再补一个加餐。这份加餐跟今天的大有不同。我会将前 10 讲的课程串联起来，精讲每一讲的主题，对内容详加梳理，列提纲、划重点（敲黑板），也就是帮你把豆子们都找出来、串起来。

当然，我需要在这里强调的事情是，这件事情一做过，就意味着“你自己找豆子”就结束了。豆子是你找来的，还是我拿给你看的，大不相同。

所以我觉得啊，你还是自己多努力找找。如果你需要补补课，加强一些基础概念方面的知识，那么我希望你有时间读一下 [🔗《程序原本》](#)，限于这节课要讲的内容，你只需要读一下《程序原本》前 10 章的内容就可以了，并且，有许多内容可以跳过去。是的，即使不懂、“不求甚解”也是可以的。有些东西就可以先“存而不论”，而这些等到你将来回头来看时，便可以立时了然。

另外，如果你的英语还不错，那么仍然推荐你看看🔗《ECMAScript 规范》，一些概念上它定义得严谨得多。不过这些概念背后的东西，就得你自己去体会了，ECMAScript 里面是不讲的。这里还有一份🔗W3C 组织的中文译本，虽然只是 ECMAScript5 的，而且还不完整，但是要达到“补概念”这个目的，还是够用了。

其他

最后呢，按照课程的习惯啊，我还是给你留个思考题。不过，跟以前不同，这道题，你答不答得出来，都是不要紧的，就算“想着玩儿”就好了。问题是这样的：

按照前面说的，所谓“会吃”，有三件事情，是食材、味道和“懂”这一个字儿。食材，我们讲了，是课程的内容；味道，我们也讲了，是课程中的教与学的法子，以及“形成体系性”这样的潜在目的。那么，什么是“懂”呢？

这个问题，就留给你了。我想啊，要知道什么是“懂”，大概才真的算是“会吃”了。

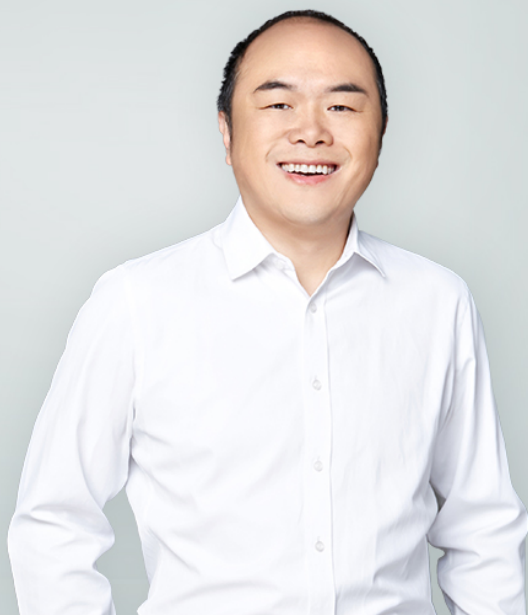
我今天的课程就聊到这里。希望你吃得好，胃口好，好好消化这一份专属的加餐，然后我们下一讲，再继续学习。

点击参与👉

打卡46天，彻底搞定 JavaScript



扫一扫参与小程序打卡



新版升级：点击「👤请朋友读」，20位好友免费读，邀请订阅更有**现金**奖励。

精选留言 (13)

写留言



潇潇雨歇

2019-11-27

我这样理解的，懂呢，就是知道要怎么吃，什么好吃，为什么这么吃好吃，对于食材和味道都很讲究，好的食材，好的味道，然后加上自己懂，那就是会吃了。

展开



3



Smallfly

2019-11-27

学习过程中的确有很多内容与已有的知识体系冲突，又找不出原因，看到老师的加餐又重拾了学习的信心，准备从第一讲开始重来过.....

展开



3



TongObama

2019-11-29

授人以鱼不如授人以渔，“捡豆子”这件事儿就是教我们自己“钓鱼”。好的老师会告诉你考点让你的分数有所谓的增加，像爱民老师这样的老师（我心目中最优秀的老师之一）会教给你如何独立去思考，以便自己能够独立找到隐藏在考点后面的秘密。所谓“懂”，我理解是建立自己的认知与思维模式的架构，无论是学习JavaScript还是任何东西都可以...

展开



1



Marvin

2019-11-28

用其长避其短，放下自己的功利心，还原其本来的味道，并将之发挥到极致。这里的“懂”这个字，我觉得很形而上了。把目标看成朋友，认真了解和相处，对其负责，这种心态可能操作性更强。

作者回复: 🙏



1



kittyE

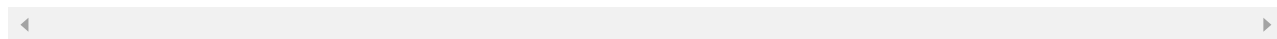
2019-11-28

老师，小学数学老师对您影响很大吗？

展开 ∨

作者回复: 一是我对数学的兴趣起源于小学；二是这位小学数学老师非常好，值得尊敬；三是那本书中，所用到的数学知识，止于小学足矣。

:)



1



三叶草

2019-11-28

假设在去捡西瓜路上，也不是一下就能捡起西瓜，而是经过不断锻炼，也是通过捡豆子开始，一步步让自己有能力去捡西瓜。大道至简，悟性无法控制，但是努力和坚持可以控制。一遍不理解那就两遍



zcdll

2019-11-28

多问几个“为什么”！

我理解的“懂”应该是除了基本的，要懂为什么要用这个食材，为什么用这个食材会产生这个味道，为什么要产生这个味道。

多问“为什么”

展开 ∨



佳民

2019-11-27

我对懂的理解是，能把握核心，洞察本质，例如每篇文章标题中的语句对应的原理是什么，又如这个专栏是编程语言层面知识，而不是工程使用的内容。我理解的懂吃是知道菜有哪些做法，为什么要这么做，以及应该配什么佐料和辅材。

展开 ∨



MarlboroKay

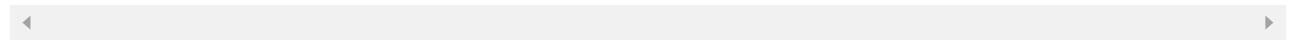
2019-11-27

$a.x = a = \{n:2\}$ 这讲到目前为止读了三遍，每次都会有新的收获，像是挖宝藏一样；还记得一年前读《你不知道的JavaScript》上，中卷，当时的感觉就是哇，原来是这样，原来还可以这么操作。

读老师的课程，更多的会问自己为什么是这样，返列是什么样。反正豆子就在那里，要靠自己的努力去捡起来。

展开 ▾

作者回复: 赞的! ^^.



许童童

2019-11-27

懂就是形成自己吃的体系，换一个食材，可以自己品出味道，好味道坏味道，以及未知的味道，甚至不吃也知道什么味道

展开 ▾



穿秋裤的男孩

2019-11-27

破而后立?

展开 ▾



2019

2019-11-27

我这个后端都心动了

展开 ▾



Mr_Liu

2019-11-27

懂得如何用现有‘食材’‘吃’出它的真实味道，如何用这种方式去吃其他‘食材’

