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。

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。

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。

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。

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。

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。

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。

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。

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。

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。

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。

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。

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。

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。

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。

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。