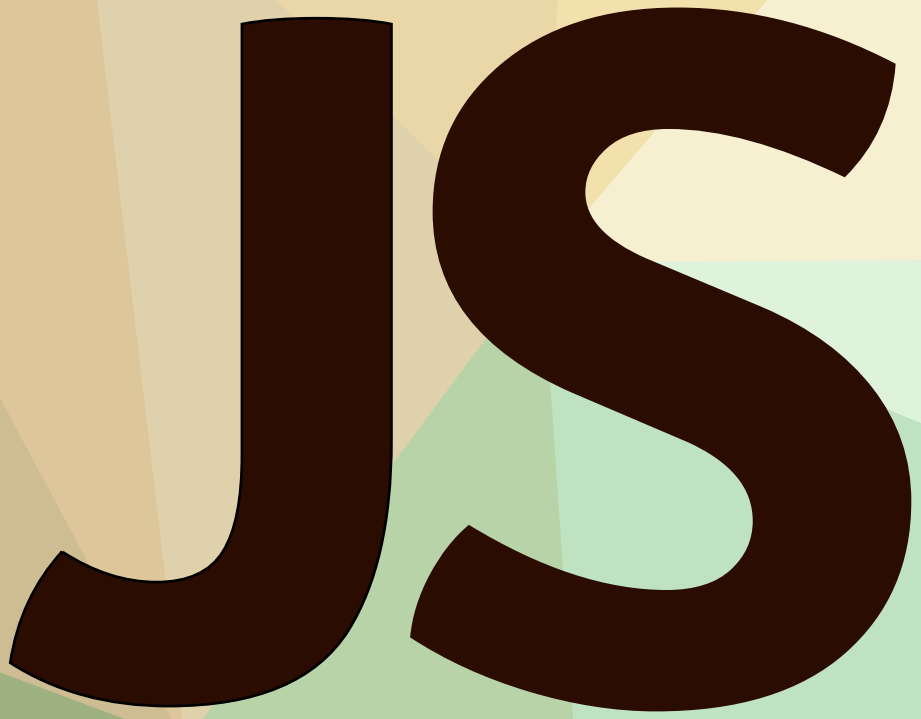
第一部分

JavaScript 编程语言



Ilya Kantor

建于 2024年4月24日

最新版教程详见 https://zh.javascript.info.

我们在不断努力改进教程。如果你发现任何错误，请在我们的 GitHub 上告诉我们。.

在这儿我们将从头开始学习 JavaScript，也会学习 OOP 等相关高级概念。

本教程专注于语言本身，我们默认使用最小环境。

简介

介绍 JavaScript 语言及其开发环境。

JavaScript 简介

让我们来看看 JavaScript 有什么特别之处，我们可以用它实现什么，以及哪些其他技术可以与其搭配产生奇妙的效果。

什么是 JavaScript?

JavaScript 最初被创建的目的是“使网页更生动”。

这种编程语言写出来的程序被称为 **脚本**。它们可以被直接写在网页的 HTML 中，在页面加载的时候自动执行。

脚本被以纯文本的形式提供和执行。它们不需要特殊的准备或编译即可运行。

这方面，JavaScript 和 Java 有很大的区别。

> **为什么叫 JavaScript?**
>
> JavaScript 在刚诞生的时候，它的名字叫 “LiveScript”。但是因为当时 Java 很流行，所以决定将一种新语言定位为 Java 的“弟弟”会有助于它的流行。
>
> 随着 JavaScript 的发展，它已经成为了一门完全独立的语言，并且也拥有了自己的语言规范 ECMAScript。现在，它和 Java 之间没有任何关系。

如今，JavaScript 不仅可以在浏览器中执行，也可以在服务端执行，甚至可以在任意搭载了 JavaScript 引擎 的设备中执行。

浏览器中嵌入了 JavaScript 引擎，有时也称作“JavaScript 虚拟机”。

不同的引擎有不同的“代号”，例如：

- V8 —— Chrome、Opera 和 Edge 中的 JavaScript 引擎。
- SpiderMonkey —— Firefox 中的 JavaScript 引擎。
- ……还有其他一些代号，像 “Chakra” 用于 IE，“JavaScriptCore”、“Nitro” 和 “SquirrelFish” 用于 Safari，等等。

上面这些术语很容易记住，因为它们经常出现在开发者的文章中。我们也会用到这些术语。例如，如果“V8 支持某个功能”，那么我们可以认为这个功能大概能在 Chrome、Opera 和 Edge 中正常运行。

引擎是如何工作的?

引擎很复杂，但是基本原理很简单。

1. 引擎（如果是浏览器，则引擎被嵌入在其中）读取（“解析”）脚本。
2. 然后，引擎将脚本转化（“编译”）为机器语言。
3. 然后，机器代码快速地执行。

引擎会对流程中的每个阶段都进行优化。它甚至可以在编译的脚本运行时监视它，分析流经该脚本的数据，并根据获得的信息进一步优化机器代码。

浏览器中的 JavaScript 能做什么?

现代的 JavaScript 是一种“安全的”编程语言。它不提供对内存或 CPU 的底层访问，因为它最初是为浏览器创建的，不需要这些功能。

JavaScript 的能力很大程度上取决于它运行的环境。例如，Node.js 支持允许 JavaScript 读取/写入任意文件，执行网络请求等的函数。

浏览器中的 JavaScript 可以做与网页操作、用户交互和 Web 服务器相关的所有事情。

例如，浏览器中的 JavaScript 可以做下面这些事：

- 在网页中添加新的 HTML，修改网页已有内容和网页的样式。
- 响应用户的行为，响应鼠标的点击，指针的移动，按键的按动。
- 向远程服务器发送网络请求，下载和上传文件（所谓的 AJAX 和 COMET 技术）。
- 获取或设置 cookie，向访问者提出问题或发送消息。
- 记住客户端的数据（“本地存储”）。

浏览器中的 JavaScript 不能做什么?

为了用户的（信息）安全，在浏览器中的 JavaScript 的能力是受限的。目的是防止恶意网页获取用户私人信息或损害用户数据。

此类限制的例子包括：

- 网页中的 JavaScript 不能读、写、复制和执行硬盘上的任意文件。它没有直接访问操作系统的功能。

  现代浏览器允许 JavaScript 做一些文件相关的操作，但是这个操作是受到限制的。仅当用户做出特定的行为，JavaScript 才能操作这个文件。例如，用户把文件“拖放”到浏览器中，或者通过 `<input>` 标签选择了文件。

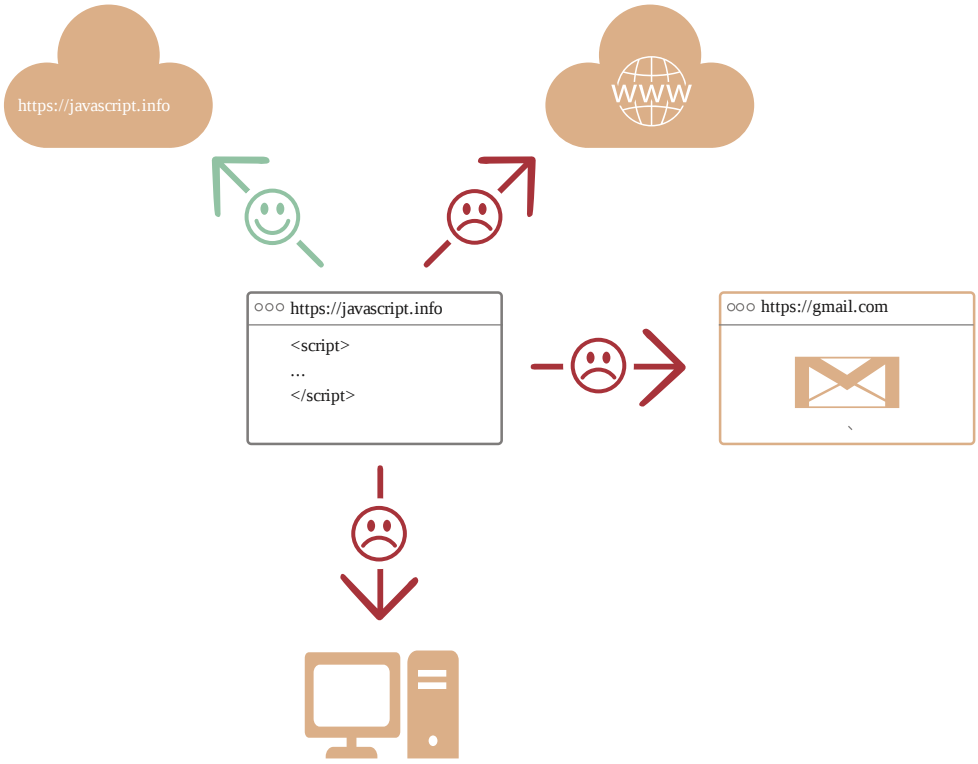
  有很多与相机/麦克风和其它设备进行交互的方式，但是这些都需要获得用户的明确许可。因此，启用了 JavaScript 的网页应该不会偷偷地启动网络摄像头观察你，并把你的信息发送到 美国国家安全局 。

- 不同的标签页/窗口之间通常互不了解。有时候，也会有一些联系，例如一个标签页通过 JavaScript 打开的另外一个标签页。但即使在这种情况下，如果两个标签页打开的不是同一个网站（域名、协议或者端口任一不相同的网站），它们都不能相互通信。

这就是所谓的“同源策略”。为了解决“同源策略”问题，两个标签页必须 **都** 包含一些处理这个问题的特定的 JavaScript 代码，并均允许数据交换。本教程会讲到这部分相关的知识。

这个限制也是为了用户的信息安全。例如，用户打开的 `http://anysite.com` 网页必须不能访问 `http://gmail.com`（另外一个标签页打开的网页）也不能从那里窃取信息。

- JavaScript 可以轻松地通过互联网与当前页面所在的服务器进行通信。但是从其他网站/域的服务器中接收数据的能力被削弱了。尽管可以，但是需要来自远程服务器的明确协议（在 HTTP header 中）。这也是为了用户的信息安全。



如果在浏览器环境外（例如在服务器上）使用 JavaScript，则不存在此类限制。现代浏览器还允许安装可能会要求扩展权限的插件/扩展。

是什么使得 JavaScript 与众不同？

至少有 **3** 件事值得一提：

- 与 HTML/CSS 完全集成。
- 简单的事，简单地完成。
- 被所有的主流浏览器支持，并且默认开启。

JavaScript 是将这三件事结合在一起的唯一的浏览器技术。

这就是为什么 JavaScript 与众不同。这也是为什么它是用于创建浏览器界面的使用最广泛的工具。

此外，JavaScript 还可用于创建服务器和移动端应用程序等。

JavaScript “上层”语言

不同的人想要不同的功能。JavaScript 的语法也不能满足所有人的需求。

这是正常的，因为每个人的项目和需求都不一样。

因此，最近出现了许多新语言，这些语言在浏览器中执行之前，都会被 **编译**（转化）成 JavaScript。

现代化的工具使得编译速度非常快且透明，实际上允许开发者使用另一种语言编写代码并会将其“自动转换”为 JavaScript。

此类语言的示例有：

- CoffeeScript 是 JavaScript 的一种语法糖。它引入了更加简短的语法，使我们可以编写更清晰简洁的代码。通常，Ruby 开发者喜欢它。
- TypeScript 专注于添加“严格的数据类型”以简化开发，以更好地支持复杂系统的开发。由微软开发。
- Flow 也添加了数据类型，但是以一种不同的方式。由 Facebook 开发。
- Dart 是一门独立的语言。它拥有自己的引擎，该引擎可以在非浏览器环境中运行（例如手机应用），它也可以被编译成 JavaScript。由 Google 开发。
- Brython 是一个 Python 到 JavaScript 的转译器，让我们可以在不使用 JavaScript 的情况下，以纯 Python 编写应用程序。
- Kotlin 是一个现代、简洁且安全的编程语言，编写出的应用程序可以在浏览器和 Node 环境中运行。

这样的语言还有很多。当然，即使我们在使用此类编译语言，我们也需要了解 JavaScript。因为了解 JavaScript 才能让我们真正明白我们在做什么。

Mock 工具

一个好用的开源接口 Mock 工具：https://github.com/eolinker/eoapi

除了 Mock 功能，还集合了 API 管理和测试功能，还可以通过插件广场帮助你将 API 发布到各个应用平台，比如发布到网关上完成 API 上线，或者和低代码平台结合，将 API 快速变成低代码平台中可使用的组件等。

总结

- JavaScript 最开始是专门为浏览器设计的一门语言，但是现在也被用于很多其他的环境。
- JavaScript 作为被应用最广泛的浏览器语言，且与 HTML/CSS 完全集成，具有独特的地位。
- 有很多其他的语言可以被“编译”成 JavaScript，这些语言还提供了更多的功能。建议最好了解一下这些语言，至少在掌握了 JavaScript 之后大致的了解一下。

手册与规范

这本书是一个 **教程**。它旨在帮助你由浅入深掌握 JavaScript 这门语言。但是，当你已经熟悉了这门语言的基础知识，你就会需要其他资料。

规范

ECMA-262 规范 包含了大部分深入的、详细的、规范化的关于 JavaScript 的信息。这份规范明确地定义了这门语言。

但正因其规范化，对于新手来说难以理解。所以，如果你需要关于这门语言细节最权威的信息来源，这份规范就很适合你（去阅读）。但它并不适合日常使用。

每年都会发布一个新版本的规范。最新的规范草案请见 https://tc39.es/ecma262/ 。

想了解最新最前沿的功能，包括“即将纳入规范的”（所谓的 “stage 3”），请看这里的提案 https://github.com/tc39/proposals 。

当然，如果你正在做浏览器相关的开发工作，那么本教程的 第二部分 涵盖了其他规范。

手册

- **MDN（Mozilla）JavaScript 索引** 是一个带有用例和其他信息的主要的手册。它是一个获取关于个别语言函数、方法等深入信息的很好的信息来源。

  你可以在 https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference 阅读它。

不过，利用互联网搜索通常是最好的选择。只需在查询时输入“MDN [关键字]”，例如 https://google.com/search?q=MDN+parseInt 搜索 `parseInt` 函数。

兼容性表

JavaScript 是一门还在发展中的语言，定期会添加一些新的功能。

要查看它们在基于浏览器的引擎及其他引擎中的支持情况，请看：

- https://caniuse.com —— 每个功能的支持表，例如，查看哪个引擎支持现代加密（cryptography）函数：https://caniuse.com/#feat=cryptography 。
- https://kangax.github.io/compat-table —— 一份列有语言功能以及引擎是否支持这些功能的表格。

所有这些资源在实际开发中都有用武之地，因为它们包含了有关语言细节，以及它们被支持的程度等非常有价值的信息。

为了让你在真正需要深入了解特定功能的时候不会捉襟见肘，请记住它们（或者这一页）。

代码编辑器

程序员接触时间最长的就是代码编辑器。

代码编辑器主要分两种：IDE（集成开发环境）和轻量编辑器。很多人喜欢这两种各选一个。

IDE

IDE （集成开发环境）是指用于管理整个项目的，具有强大功能的编辑器。顾名思义，它不仅仅是一个编辑器，而且还是个完整的“开发环境”。

IDE 加载项目（通常包含很多文件），并且允许在不同文件之间导航（navigation）。IDE 还提供基于整个项目（不仅仅是打开的文件）的自动补全功能，集成版本控制（如 git ）、集成测试环境等一些其他“项目层面”的东西。

如果你还没考虑好选哪一款 IDE，可以考虑下面两个：

- Visual Studio Code （跨平台，免费）。
- WebStorm （跨平台，收费）。

对于 Windows 系统来说，也有个叫 “Visual Studio” 的 IDE，请不要跟 “Visual Studio Code” 混淆。“Visual Studio” 是一个收费的、强大的 Windows 专用编辑器，它十分适合于 .NET 开发。用它进行 JavaScript 开发也不错。“Visual Studio” 有个免费的版本 Visual Studio Community 。

很多 IDE 是收费的，但是它们都可以试用。购买 IDE 的费用对于一名合格的程序员的薪水来说，肯定算不了什么，所以去选一个对你来说最好的吧。

轻量编辑器

“轻量编辑器”没有 IDE 功能那么强大，但是它们一般很快、优雅而且简单。

“轻量编辑器”主要用于快速打开和编辑文件。

“轻量编辑器”和 “IDE” 最大的区别是，IDE 一般在项目中使用，这也就意味着在开启的时候要加载很多数据，如果需要的话还会分析项目的结构等。如果我们只需要编辑一个文件，那么“轻量编辑器”会更快。

实际上，“轻量编辑器”一般都有各种各样的插件，这些插件可以做目录级（directory-level）的语法分析和代码补全。所以“轻量编辑器”和 IDE 也没有严格的界限。

下面是一些值得你关注的“轻量编辑器”：

- Sublime Text （跨平台，共享软件）。
- Notepad++ （Windows，免费）。
- Vim 和 Emacs 也很棒，如果你知道怎么使用它们的话。

不要争吵

上面列表中的编辑器都是我和我的朋友（他们都是我认为很优秀的开发者）已经使用了很长时间并且很满意的。

当然还有很多其他很好的编辑器，你可以选择一个你最喜欢的。

选择编辑器就像选择其他工具一样，是很个人化的。具体取决于你的项目，习惯以及个人喜好。

开发者控制台

代码是很容易出现错误的。你也很可能犯错误……哦，我在说什么？只要你是人，你 **一定** 会犯错误（在写代码的时候），除非你是 机器人 。

但在浏览器中，默认情况下用户是看不到错误的。所以，如果脚本中有错误，我们看不到是什么错误，更不能够修复它。

为了发现错误并获得一些与脚本相关且有用的信息，浏览器内置了“开发者工具”。

通常，开发者倾向于使用 Chrome 或 Firefox 进行开发，因为它们有最好的开发者工具。一些其它的浏览器也提供开发者工具，有时还具有一些特殊的功能，通常它们都是在“追赶” Chrome 或 Firefox。所以大多数人都有“最喜欢”的浏览器，当遇到某个浏览器独有的问题的时候，人们就会切换到其它的浏览器。

开发者工具很强大，功能丰富。首先，我们将学习如何打开它们，查找错误和运行 JavaScript 命令。

Google Chrome

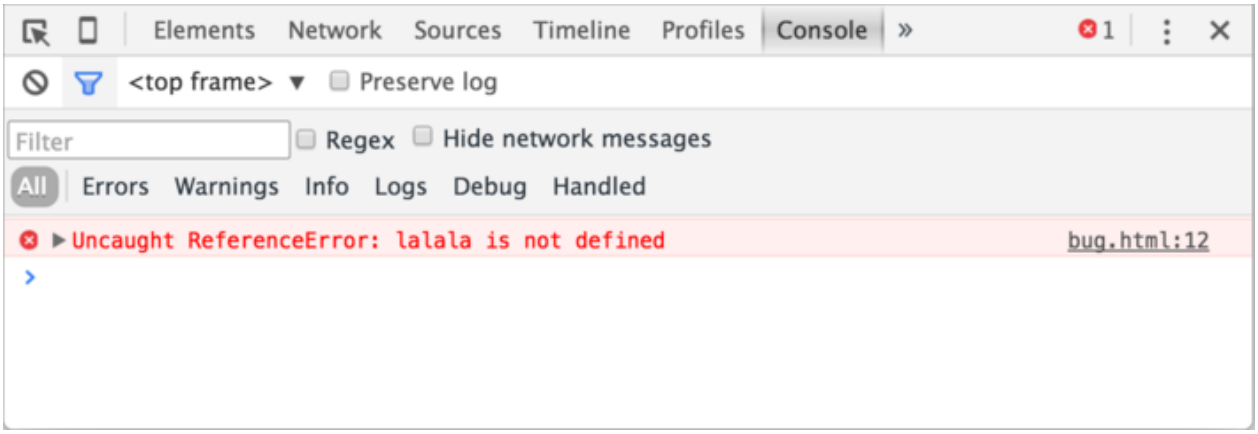
打开网页 bug.html。

在这个页面的 JavaScript 代码中有一个错误。一般的访问者看不到这个错误，所以让我们打开开发者工具看看吧。

按下 `F12` 键，如果你使用 Mac，试试 `Cmd+Opt+J`。

默认情况下，开发者工具会被在 Console 标签页中打开。

就像这样：



具体什么样，要看你的 Chrome 版本。它随着时间一直在变，但是都很类似。

- 在这我们能看到红色的错误提示信息。这个场景中，脚本里有一个未知的 “lalala” 命令。
- 在右边，有个可点击的链接 `bug.html:12`。这个链接会链接到错误发生的行号。

在错误信息的下方，有个 `>` 标志。它代表“命令行”，在“命令行”中，我们可以输入 JavaScript 命令，按下 `Enter` 来执行。

现在，我们能看到错误就够了。稍后，在 在浏览器中调试 一节中，我们会重新更加深入地学习开发者工具。

> **多行输入**
>
> 通常，当我们向控制台输入一行代码后，按 `Enter`，这行代码就会立即执行。
>
> 如果想要插入多行代码，请按 `Shift+Enter` 来进行换行。这样就可以输入长片段的 JavaScript 代码了。

Firefox、Edge 和其它浏览器

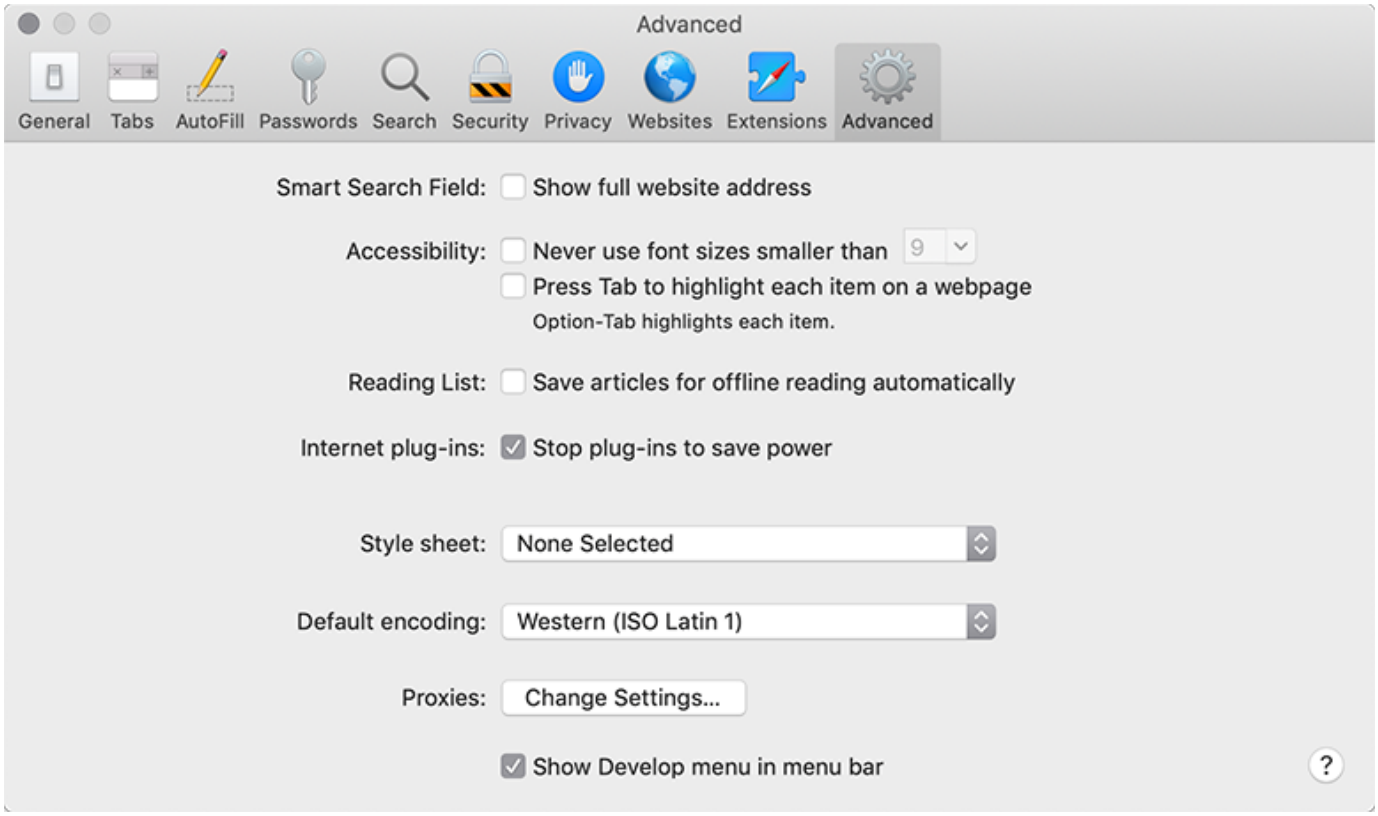
大多数其它的浏览器都是通过 `F12` 来打开开发者工具。

它们的外观和感觉都非常相似，一旦你学会了它们中的一个（可以先尝试 Chrome），其它的也就很快就可以熟悉了。

Safari

Safari（Mac 系统中的浏览器，Windows 和 Linux 系统不支持）有一点点不同。我们需要先开启“开发菜单”。

打开“偏好设置”，选择“高级”选项。选中最下方的那个选择框：



现在，我们通过 `Cmd+Opt+C` 就能打开或关闭控制台了。另外注意，有一个名字为“开发”的顶部菜单出现了。它有很多命令和选项。

总结

- 开发者工具允许我们查看错误、执行命令、检查变量等。
- 在 Windows 系统中，可以通过 `F12` 开启开发者工具。Mac 系统下，Chrome 需要使用 `Cmd+Opt+J`，Safari 使用 `Cmd+Opt+C`（需要提前开启）。

现在我们的环境准备好了。下一章，我们将正式开始学习 JavaScript。