Javascript

JavaScript 最初被创建的目的是“处理表单验证”，后来我们更多应用与它使网页更生动

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”）。

关于Javascript

Javascript是用于创建浏览器界面的使用最广泛的工具
在Web世界里，只有JavaScript能跨平台、跨浏览器驱动网页，还与用户交互。

- 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。

随着HTML5在PC和移动端越来越流行，JavaScript变得更加重要了。并且，新兴的Node.js把JavaScript引入到了服务器端，JavaScript已经变成了全能型选手。

Javascript发展史

Netscape网景公司 布兰登.艾奇 10天时间搞出来了JavaScript，作为控制浏览器和给网页添加活力和交互性的方法；以前叫 LiveScript；
后来网景公司被Sun公司收购，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，所以改名为JavaScript。

但是，它和 Java 之间没有任何关系

> javascript，是一种**弱类型的脚本语言**，支持多范式开发(面向过程，但是也可以实现面向对象)。

什么是语言

- 计算机就是一个由人来控制的机器，人让它干嘛，它就得干嘛。
- 我们要学习的语言就是人和计算机交流的工具，人类通过语言来控制、操作计算机。
- 编程语言和我们说的中文、英文本质上没有区别，只是语法比较特殊。
- 语言的发展：
    - 编程语言分为3个历史阶段:

1. **机器语言**：直接使用由“0”和“1”组成的二进制指令控制计算机，二进制是计算机的语言的基础。

用计算机的语言去命令计算机干这干那，一句话，就是写出一串串由“0”和“1”组成的指令序列交由计算机执行，这种语言，就是机器语言。
使用机器语言是十分痛苦的，特别是在程序有错需要修改时，更是如此。而且，由于每台计算机的指令系统往往各不相同，
所以，在一台计算机上执行的程序，要想在另一台计算机上执行，必须另编程序，造成了重复工作。但由于使用的是针对特定型号计算机的语言，故而运算效率是所有语言中最高的。机器语言，是第一代计算机语言。

2. **汇编语言**：把CPU指令表示为单词或单词的缩写，极大的方便了程序员对各种指令的记忆。一定程度上简化了编程复杂度，减低了学习编程语言的成本。

   为了减轻使用机器语言编程的痛苦，人们进行了一种有益的改进：用一些简洁的英文字母、符号串来替代一个特定的指令的二进制串，
   比如，用“A D D”代表加法，“M O V”代表数据传递等等，这样一来，人们很容易读懂并理解程序在干什么，纠错及维护都变得方便了，
   这种程序设计语言就称为汇编语言，即第二代计算机语言。然而计算机是不认识这些符号的，这就需要一个专门的程序，
   专门负责将这些符号翻译成二进制数的机器语言，这种翻译程序被称为汇编程序。
   汇编语言同样十分依赖于机器硬件，移植性不好，但效率仍十分高，针对计算机特定硬件而编制的汇编语言程序，
   能准确发挥计算机硬件的功能和特长，程序精炼而质量高，所以至今仍是一种常用而强有力的软件开发工具。

3. **高级语言**：第三代编程语言，对CPU指令进行了高度的封装，使用更接近于人类语言的语法规则，编出的程序能在所有机器上通用。使编程复杂度和学习难度进一步降低。高级语言同样不能直接运行，
   需要使用编译程序将高级语言的源代码编译为汇编语言，然后才能执行。(例如c语言，某台设置只要能够运行c语言的编译器，就能够运行c语言程序)。

如何理解高级语言：

1. 首先，按照运行原理，分为编译型和解释型(脚本型)。
   - 编译型：在程序执行之前，有一个单独的编译过程，将程序的所有源代码都编译为可执行代码（机器语言），再执行编译之后的可执行代码。(c,c++,java)
   - 解释型：程序在运行的过程中：一边编译，一边执行（编译一行，执行一行）(javascript,python,php,c#)。

   理论上，编译型语言比解释型语言执行效率高。

2. 按照数据类型是否强制，可以分为强类型语言和弱类型语言。
   - 强类型：程序中变量、参数、函数返回值，都必须指明类型，不同类型的变量不能相互赋值。(c,java,c++,c#)
   - 弱类型：变量、参数、函数返回值不需要指定类型，一个变量的类型是不固定的。(js,vb)

3. 按照编程范式，可分为面向对象语言和面向过程语言。
   - 面向过程：将函数作为程序的基本单元，着重程序流程的开发。(c,js,pascal,vb)。
   - 面向对象：类和对象是程序的基本单元，函数不能独立存在，着重类和对象的开发。(c++,java,objective-C)。

JavaScript 组成

1. **ECMAScript** - JavaScript的核心

ECMA 欧洲计算机制造联合会

网景：JavaScript

微软：JScript

定义了JavaScript的语法规范

JavaScript的核心，描述了语言的基本语法和数据类型，ECMAScript是一套标准，定义了一种语言的标准与具体实现无关 、ECMAScript是由Ecma国际标准组织以ECMA-262和ECMA-402规范的形式进行标准化的；

2. **BOM** - 浏览器对象模型

一套操作浏览器功能的API

通过BOM可以操作浏览器窗口，比如：弹出框、控制浏览器跳转、获取分辨率等

3. **DOM** - 文档对象模型

一套操作页面元素的API

DOM可以把HTML看做是文档树，通过DOM提供的API可以对树上的节点进行操作

深入了解ES与JS的关系

一个常见的问题是，ECMAScript 和 JavaScript 到底是什么关系？

ES是JS的规范，JS是ES的实现；

要讲清楚这个问题，需要回顾历史。

- JavaScript诞生于1995年，它的出现主要是用于处理网页中的前端验证。
- 所谓的前端验证，就是指检查用户输入的内容是否符合一定的规则
- 比如：用户名的长度，密码的长度，邮箱的格式等。
- 1996年微软公司在其最新的IE3浏览器中引入了自己对 JavaScript的实现JScript。
- 于是在市面上存在两个版本的JavaScript，一个网景公司的 JavaScript和微软的JScript。
- 为了确保不同的浏览器上运行的JavaScript标准一致，所以几 个公司共同定制了JS的标准名命名为ECMAScript。简称ES，每年都在更新，目前最火的是ES6

> 拓展：JS的执行
>
> 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 和 Opera 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，“ChakraCore” 用于 Microsoft Edge，“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 和 Opera 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