# JavaScript 目录

## 一：简介

首先：这不是JavaScript各个知识点的详细叙述、也不是JavaScript教程。只是自己在学习JavaScript时的学习笔记。作用有二：一是加深自己对学习过程、内容的理解。二是方便以后回顾翻阅、自己不是那种学一遍就掌握的天才、温故而知新。

对JavaScript的从基础入门到深入的一些整理。可以说自己会JavaScript、平时敲代码必不可少的东西。也可以说不会、静下心来想想、关于JavaScript的印象仅限于函数、事件、获取元素的方法、js脚本这些散件。毫无组织可言、简而言之：对JavaScript的认知只是片面性的。在此则希望系统的从基础到高级的系统的学习、梳理一下JavaScript知识、方便建立一个完善的知识体系。

但是不会具体的像书本教程一样的面面俱到的记录、解释所有知识点。有时候有可能只是会列出类似目录提纲一样的东西、偶尔会对一些知识点进行多一点的记录。此笔记的核心是通过各个知识点将JavaScript的整体结构记录在脑海中。目的是当看到这些知识点时、知道他们具体的功能与意义进而加深对JavaScript的理解与掌握。便于进一步在脑海中建立一个完善的知识库。

如果有意了解没有详述的知识点具体内容则可以google、百度。此笔记内容实例都经过自己测试、整理至网络和《JavaScript DOM编程艺术》第二版（英文名：DOM Scripting – Web Design with JavaScript and the Document Object Model）。有兴趣的可以看看这本书。

## 二：目录

JavaScript 简史

JavaScript 基础语法

JavaScript 数据类型

JavaScript DOM

JavaScript DOM Core、 HTML DOM与CSS DOM

## TO BE CONTINUED…

# JavaScript 简史

## 一：JavaScript起源

JavaScript是Netscape与Sun一起合作开发的。其第一个版本出现在1995年退出的Netscape Navigator 2 浏览器中。

JavaScript与Java语言没有任何关系。Java理论上可以部署在任何环境、而JavaScript是一种脚本语言、只应用于Web浏览器。

JavaScript向程序员提供了一些操作Web浏览器的手段。

## 二：DOM

DOM（Document Object Model）：是一套对文档内容进行抽象和概念化的方法。后面会有详述。

## 三：浏览器战争

一言而蔽之：各个浏览器对DOM的解析都有一套自己的标准、导致相同的操作或者效果在不同的浏览器中需要不同的实现方式。

### DHTML

Dynamic HTML（动态HTML）用于描述HTML、CSS、JavaScript技术组合的术语。其背后含义：

1. 利用HTML把网页标记为各种元素
2. 利用CSS设置元素样式和他们的显示位置
3. 利用JavaScript实时操控页面和改变样式。

### 浏览器之间的冲突

不同浏览器对DOM操作支持不同、同样的JavaScript脚本在不同的浏览器中不一定全部都能正确执行。

## 四：制定标准

W3C（World Wide Web Consortium）联合浏览器制造商指定标准化DOM。DOM是一种API。其定义：一个与系统平台和编程语言无关的接口、程序和脚本可以通过这个接口动态地访问和修改文档的内容、结构和样式。

如今几乎所有的浏览器都内置了对DOM标准的支持、实现“一次编写，随处运行”梦想。

## 五：DOM、JavaScript以及jQuery之间的关系

DOM(document object model) 是一套对文档内容进行抽象和概念化的方法、是对外提供的操纵浏览器内容的API。

JavaScript对网页进行的所有操作都是通过DOM进行的、用于改变浏览器呈现的内容与方式。DOM属于浏览器，而不是JavaScript语言规范里的规定的核心内容。

jQuery是继prototype之后又一个优秀的JavaScript框架、是JavaScript库。

# JavaScript 语法

## 一：简介

JavaScript是一种解释性程序设计语言、不需要编译器、只需要解释器。Web浏览器负责完成有关的解释和执行工作。

编译JavaScript不需要任何特殊工具（有则更加、比如jetbrains webstorm，Sublime Text等神器）。

页面中使用JavaScript：1、在页面中任何位置添加<script> //body…</script>. 2、者将所有的JavaScript内容放在一个文件名为\*\*\*.js的文件中、在页面中使用<script src=’\*\*\*.js’></script>引入。

## 二：语法

### 语句

用JavaScript编写的脚本都由一系列指令构成、这些指令叫做语句（statement）。多条语句可以放在一行、之间用分号分割。可以不用分号、每条语句单独一行。建议一条语句放一行、并且用分号结束。好的编程习惯是基本素养。

### 注释

用于帮助理解程序执行语句的文档型信息、这些信息会被解释器忽略。常用注释方式：

//单行注释

/\* 块注释 \*/

/\*\*文档型注释、显示相关类文档时显示、Java类中很常见\*/

<!—HTML风格的注释、可单行可块。 -->

### 变量

变量（variable）：会发生变化的东西。

变量相关：声明（declare）、赋值（assignment）。JavaScript是弱类型（weakly typed）语言、其变量定义不管变量是什么类型一律使用var做前缀。如var name = "alien", age = 25;

变量的作用域：

1. 全局变量（global variable）：作用于整个脚本、在任何函数（function、下面有定义）外部定义。
2. 局部变量（local variable）：只作用于声明他的那个函数内部。

### 数据类型

JavaScript中数据类型分六种——五种基本类型、一种引用类型。具体如下：

基本类型：boolean、number、string、null、undefined。

引用类型：Object。Object又分user-defined object。Native object（如Date、Math等内置对象）。Array、function这些在JavaScript中都属于Object类型。

也可做如下划分：

主数据类型：String、Number、Booelan

符合数据类型：Object、Array

特定数据类型：Null、undefined

数据类型与下面的操作符结合、以及如何判断一个变量是哪种类型时会有一些容易让人疑惑的地方、会在下一笔记中单独记录。这里只是概念性记录。

## 三：操作

算术操作符（arithmetic operation）：+ - \* / += -=

自增、自减（分操作前、操作后）：++ --

比较操作符（comparison operators）：> < >= <= == != ===

逻辑操作符（logical operators）：|| && !

## 四：条件语句

if (condition) {  
 statements;  
}else if (condition) {  
 statements;  
}else{  
 statements;  
}

## 五：循环语句

### while循环

//while circulation  
while (condition) {  
 statements;  
}

### do … while 循环

do{  
 statements;  
}while(condition);

### for循环

for (initial condition; test condition; alter condition) {  
 statements;  
}

## 六：函数

把需要多次使用的同一段代码封装成一个函数、函数（function）就是一组允许在代码里随时调用的语句。

function methodName ( parameter ... ) {  
 //body statements;  
}

## 七：对象

### 对象的属性与方法：

重复对象这种数据类型是因为其重要性。对象是自包含的数据集合、包含在对象里的数据可以通过两种形式访问——属性（property）和方法（method）：

属性是隶属于某个对象的变量：Object.property

方法只有某个特定对象才能调用的函数：Object.method()

### 对象分类

1. 内建对象（native object）: JavaScript提供的一系列预先定义好的对象。如数组就是其中之一、还有常用的Math、Date。这些对象是拿来就可以直接使用的。
2. 宿主对象（host object）: 浏览器提供的预定义的对象。如Form、Image、Element、document对象等。
3. 用户自定义对象（user-defined object）：用户自定义对象、定义方式：function Person(){ //body… }。

# JavaScript 数据类型

## 一：简介

再提数据类型是因为其与操作符结合、以及如何判断一个变量是哪种类型时会有一些容易让人疑惑的地方、比如数据类型与操作符组合之后的结果有时候并不是我们想象的那样。再比如=、== 与===有什么区别。

## 二：隐式转换

下面的测试可以通过chrome浏览器的开发者模式（按F12调出）的console中直接输入、回车后再输入相应的变量名就可以很直观的看到结果。

JavaScript中比较两个数据是否相同可以使用“==”或者“===”。前者在比较时如果两个数据类型不同、则会做一定转换再比较。后则为绝对相等、如果相比较的两个对象类型不同则直接返回false。

隐式转换见于两个相比较对象对比时。

1. a===b:
2. 类型不同直接返回false、
3. 类型相同再进一步比较值是否相同、需要注意的是：如果是对象比较、JavaScript比较的是两个对象的引用是否为同一引用、是则为true、否则为false。
4. null === null、undefined === undefined、NaN != NaN、new Object != new Object.
5. a==b :
6. 类型相同、与a===b等价。
7. 类型不同
   1. null与undefined相等
   2. number与String相比时String转换成number
   3. Boolean与其他相比时、true转换成1 false转换成0.但是如果是先定义一个变量值为 1 再与true做对比则结果为false。
   4. object与其他相比时、object转为基本类型、其他false。
8. exa mple：

## 三：类型检测

JavaScript中可以通过一下几种方式来判断一个数据的类型：typeof、instanceof、Object.prototype.toString、constructor、duck type.

1. typeof:

a) 适用于判断基本类型和函数类型。

b) typeof null结果为“Object”。

c) typeof判断数组、输出结果为Object。

example：

2. instanceof

a) 适用于判断被检测类型属于哪种对象类型——obj instanceof Object;

b) 其左边的运算数是一个对象，右边的运算数是对象类的名字或构造函数。如果 object 是 class 或构造函数的实例，则 instanceof 运算符返回 true。如果 object 不是指定类或函数的实例，或者 object 为 null，则返回 false。

c) instanceof 不能跨iframe使用！不同iframe中创建的对象不会共享一个prototype。

example:

3. Object.prototype.toString.apply();

通过对象的prototype判断其具体类型、可用于判断数组类型

example:

1. constructor 、通过构造器判断类型、constructor可以被改写、不经常用
2. duck type 鸭子类型、通过类型特有属性或者方法来判断数据类型、如判断一个数据类型是否是数组可以查看此数据是否有length属性等。

## 四：总结

typeof：适合基本类型和function检测、遇到null则失效。

Object.prototype.toString.applay()判断。

instancof：适合自定义对象、也可以用来检测原生对象、在不同iframe和window之间失效。

后两种不常用、视情况而定。

## 五：补充

测试结果：

# JavaScript DOM

## 一：简介

前面已经提到过JavaScript与DOM之间的关系：

DOM(document object model) 是一套对文档内容进行抽象和概念化的方法、是对外提供的操纵浏览器内容的API。

JavaScript对网页进行的所有操作都是通过DOM进行的、用于改变浏览器呈现的内容与方式。DOM属于浏览器，而不是JavaScript语言规范里的规定的核心内容。

此篇重点在于DOM。

## 二：DOM

D——Document：当创建一个网页并把他加载到web浏览器中时、DOM自动生成。把编写的网页文档转换为一个文档对象。

O——Object：对象。比如上面提到的把网页文档转换成的document对象。JavaScript中对象分为三类（前文有详细说明）：用户定义对象（user-defined object）、内建对象（native object）、宿主对象（host object）。

M——Model or Map：某种事物的表现形式。

## 三：节点

元素节点（element node）：是DOM的原子、对应文档中元素、如div、p、li等、可以包含其他元素节点。

文本节点（text node）：总是被包含于元素节点。

属性节点（attribute node）：用来对元素做出更具体的描述。

## 四：CSS

Cascading Style Sheet：改变文档呈现形式、具有继承（inheritance）特性、即加于某元素的样式同样作用与此元素包含的所有元素、但是可以被覆盖。

## 五：DOM常用方法

1. 获取元素

用于document对象！

假如getElementsByClassName不起作用、可通过如下方式实现：

1. 操作元素属性

## 六：补充

这里提到的只是很少一部分DOM方法、不是DOM的API、其他还有很多方法没有列举。

# JavaScript 简单应用

## 一：简介

此笔记是对《JavaScript权威指南》第4章内容的概括、重点的记录。不是重现书本实例。书中大部分实例放在github上、有兴趣的也可以看这本书或者github上代码。github地址会放在JavaScript目录中。

## 二：JavaScript应用步骤

1. 编写JavaScript脚本
2. 在页面中引用：

a) 可以直接在页面中使用<script>标签将JavaScript相关内容放入其中。

b) 可以将脚本放入一个或者多个文件结尾为js的文件中、在需要使用的页面中引入。

## 三：实例简述

网页中有多个链接、不点击任何链接时、页面下方的显示位置会显示一张默认图片和默认说明、当点击某个具体链接时将下方显示位置的默认图片替换成链接定向到的图片、同时将链接的说明内容显示在指定位置。

## 四：实例实现关键点

1. 获取a链接的herf和title属性。

2. 为所有链接添加单击事件。

3. 阻止链接默认行为。

2. 修改默认图片标签的src和说明标签内的文本内容。

## 五：相关方法与解决方式

1. 阻止a默认行为：

2. 相关方法

## 六：补充

效果:

showPicture.js：

showPicture.html：

showPicture.css：

# JavaScript 最佳实践

## 一：简介

JavaScript已经成为web中不可或缺的一部分、相信绝大多数web如果禁用JavaScript、结果可想而知、甚至严重的直接导致整个网站瘫痪。

但是个人觉得JavaScript已经是某些情况下必不可少的一部分了、JavaScript最佳实践也成为了在应用JavaScript的理想与现实之中寻找一种平衡。

最佳实践的一些原则可以帮助我们理解JavaScript的精髓、帮助我们完善自己的程序设计和提高其健壮性。

## 二：关键之处

JavaScript与DOM结合是一种非常强大的组合、可以轻而易举的改变网页的行为与显示、怎么样才是最佳实践？下面是最佳实践的一些准则。

1. 是否需要使用JavaScript：

当想使用JavaScript来改变一个网页的行为时、首先要想到的是是否有必要这样做。而不是人云亦云的跟风。

2. 渐进增强（progressive enhancement）：

从核心部分开始、即从内容开始、根据内容使用标记实现良好的结构；再逐步加强这些内容。增强的工作既可以使用CSS改变呈现效果、也可以使用DOM添加各种行为、避免使用DOM添加核心内容！应该在开始就让内容成为文档编写时期的核心组成部分。

3. 平稳退化（graceful degradation）：

在正确使用JavaScript脚本的情况下、即使浏览器不支持JavaScript（这种现象几乎绝迹）或者在用户禁用浏览器执行JavaScript的功能也能正常浏览网站。也就意味着JavaScript是为了增强页面访问效果或者显示效果而存在、不参与核心功能（防止JavaScript被禁而废）。

4. 分离JavaScript：

把网页的结构和内容与JavaScript脚本的动作行为分开、Java中常常强调的解耦思想类似。将相互间的影响降到最低。比如链接上添加单击事件、这其实已经是把网页的结构内容与JavaScript脚本糅合在一起了。我们可以将其分离开来、但是现实中很少有这样做、一是代码可读性下降、另一个是使得编码变得复杂、需要一些额外编码来实现、这就是一种理想与现实的差距、也可能是自己水平不够的原因。

5. 向后兼容

确保网页中使用的JavaScript在老的浏览器中也能正常执行、不同浏览器之间或者浏览器版本之间的兼容性一直是一个让人很头疼的问题、但是随着技术的进步、这种现象也在逐渐好转。实现这一原则需要我们在使用JavaScript时需要考虑到其在各个浏览器、各个版本之间的兼容问题。

6. 性能考虑

确定脚本执行的性能最优。比如尽可能的减少DOM访问的次数、合并可以归类的JavaScript脚本、压缩脚本（加快客户端下载数度、可以使用Uglify或者在线的JS压缩）等。

## 三：改进前章实例

可以参照最佳实践来完善前面图片库的实例、不再一一叙述、给出完善后showPicture.js压缩前后的代码做个参考、完整代码在github上。

压缩前：

压缩后：

# JavaScript 动态创建标记与Ajax

## 一：简介

前面都是针对已经存在的标记进行操作、JavaScript同样可以动态创建标记并且与现有标记组合生成新的Document。同时简单的介绍了Ajax。

## 二：动态创建标记与组合

相关方法：

## 三：Ajax

Ajax（Asynchronous JavaScript And XML）异步JavaScript和XML、用于对页面请求以异步方式与服务器进行交互进而达到局部刷新页面的效果。

Ajax实现过程：

## 四：完善图片库

动态创建用于显示默认图片和指定链接的图片和title相关标记——showPicture.js:

# JavaScript 充实文档内容

## 一：简介

使用JavaScript来充实文档内容。主要目的是为现有文档创建一个“缩略语列表”、“文献来源链接”、“快捷键清单”。基本都是前面使用过的函数、没有什么要特别说明的地方。

## 二：效果图

## 三：具体内容

HTML的内容的编写可以使用Sublime Text （安装Emmet插件）、或者Jetbrain的Webstorm神器、效率提高N倍。有兴趣的可以百度谷歌Emmet简介。

example.html：

# JavaScript CSS-DOM

## 一：简介

JavaScript CSS-DOM是通过JavaScript去改变HTML-DOM的显示形式、即样式。如element.style.fontSize = 15em;此种形式。与CSS相比、CSS-DOM可以定时重复的执行一些样式渲染或者改变、这是CSS所不能实现的。

## 二：DOM Core HTML-DOM CSS-DOM关系

DOM Core是DOM API提供的用于操作页面元素的基础方法、比如根据id、标签名获取元素标签等

HTML-DOM 同样是JavaScript用于操作HTML元素的方法、只是方式有所不同、如document.body;element.href = value;等。

CSS-DOM是JavaScript操作页面显示样式的方式、如document.style.fontSize = 11px等。

从上面看JavaScript似乎能胜任一个网页从构造到显示、以及动态改变网页结构样式的功能。是能胜任、但是并不意味着所有的事情交由JavaScript完成是一个好的选择。

## 三：网页的三层架构

结构层（structure layer）：由HTML或XHTML之类的标记语言负责创建。

表示层（presentation layer）：由CSS负责完成、其描述网页如何显示。

行为层（behavior layer）：负责内容应该如何相应事件、由JavaScript语言和DOM完成。

分层的思想使得分工更加明确、调理更加清晰。所以、在产品设计中、选择最合适的工具去解决问题是最基本的原则。尽管上面提到的三层架构之间都有所渗透、如JavaScript可以操作三层任何一层、CSS的a:hover可以完成JavaScript类似功能等、在设计中还是应遵循选择最适合的工具去解决问题原则。

## 四：实例

展现一个表格、奇偶行背景颜色不同、当鼠标移到上面时字体变粗。

关键点：

为元素添加样式：element.style.xxx = value;

为元素绑定事件：element.onmouseover = function(){};

触发事件时改变指定元素样式。

使用了前文中的displayAbbreviations函数。

addLoadEvent.js、displayAbbreviations.js前文中有。

itinerary.html:

itinerary.css:

example.js:

# JavaScript 动画效果

## 一：简介

通过JavaScript动态改变元素位置来实现动画效果。实例、鼠标移动到链接、在下方显示对应图片。主要是网页的分层结构思想的应用。

## 二：重点

CSS中元素的position属性、overflow属性

var variable = setTimeout("function", interval);//定时触发函数

clearTimeout(variable);//清除定时触发函数、variable是setTimeout的返回值。

parseInt(str);//将字符串转换成整数

parseFloat(str);//将字符串转换成浮点数

Math.ceil(number);//向上取整

Math.floor(number);//向下取整

Math.round(number);//四舍五入

## 三：实现

效果：

list.html:

list.css:

list.js:

moveElementImprove.js:

addLoadEvent.js: