目录

[JS操作数隐式类型转换 1](#_Toc497750586)

[补码、反码、源码 1](#_Toc497750587)

[Doctype 严格模式与混杂模式的区别在哪里? 1](#_Toc497750588)

[null和undefined的区别，null == undefined的结果是？ 2](#_Toc497750589)

[sass-css预处理器 2](#_Toc497750590)

[SVG与canva区别 2](#_Toc497750591)

[ECMAScript与JavaScript的规范 3](#_Toc497750592)

[会造成内存泄漏的操作： 3](#_Toc497750593)

[CDN 3](#_Toc497750594)

[跨域 3](#_Toc497750595)

[什么是回调 4](#_Toc497750596)

[什么是同步/异步 4](#_Toc497750597)

[什么是I/O 4](#_Toc497750598)

[什么是单线程/多线程 4](#_Toc497750599)

[什么是阻塞/非阻塞 4](#_Toc497750600)

[什么是事件 4](#_Toc497750601)

[什么是事件驱动 4](#_Toc497750602)

[什么是基于事件驱动的回调 4](#_Toc497750603)

[什么是事件循环 4](#_Toc497750604)

# JS操作数隐式类型转换

JS在做==运算时会将操作数转换为Number类型之后再操作，而' '，false，0，[]都能转换为为数字类型0，而{}却转换不了，转换后为NaN,而NaN做任何运算返回都是false

**转成数值：**

[] “ ” false………0 {}…….NAN

False==null //false

**转布尔：**

Undefined null 0 NaN “”………….false [] {}……true

NaN与任何数进行这些操作，结果都是NaN

"a"-1结果为NaN，但是typeof NaN === ‘number’

1/0结果为Infinity，但是typeof Infinity === ‘number’

0 && “a” 结果为0，是number类型

new Number(1) 产生的是对象类型数据

Number() Number(0) Number('') Number('0') Number(false) Number(null) Number([]) Number([0])//0

**Number(undefined) Number({})//NaN**

console.log(null == 0); //false

console.log(undefined == 0); //false null==undefined;//true

要比较相等性之前，不能将null和undefined转换成其他任何值。就是undefined和null与其他数在进行相等判断时不进行类型转换。

null > 0   // null 尝试转型为number , 则为0 . 所以结果为 false,

null >= 0  // null 尝试转为number ,则为0 , 结果为 true.

null == 0  // null在设计上，在此处不尝试转型. 所以 结果为false.

1、如果x不是正常值（比如抛出一个错误），中断执行。

2、如果y不是正常值，中断执行。

3、如果Type(x)与Type(y)相同，执行严格相等运算x === y。

4、如果x是null，y是undefined，返回true。

5、如果x是undefined，y是null，返回true。

6、如果Type(x)是数值，Type(y)是字符串，返回x == ToNumber(y)的结果。

7、如果Type(x)是字符串，Type(y)是数值，返回ToNumber(x) == y的结果。

8、如果Type(x)是布尔值，返回ToNumber(x) == y的结果。

9、如果Type(y)是布尔值，返回x == ToNumber(y)的结果。

10、如果Type(x)是字符串或数值或Symbol值，Type(y)是对象，返回x == ToPrimitive(y)的结果。

11、如果Type(x)是对象，Type(y)是字符串或数值或Symbol值，返回ToPrimitive(x) == y的结果。

**空字符串(''),NaN,0，null,undefined**这几个外返回的都是true, 所以! []这个[] => true 取反为false,所以[] == false为true。(但遇到bool对比式先转换bool为数字，除非有！可以先转为bool)

## Number与parseInt

一般是先转为number null和undefined有特殊

({}==true)//防止{}放前面被误认成代码块； 而true=={} false

Number 不认前导符 会直接删 与parseInt认前导符 不过也提供第2个参数作为基数识别(是判断传入的第一个参数是什么进制的而不是说要转为什么进制) 不过parseFloat也不认前导符号直接去前导或是返回0 还会识别科学计数法e7

js提供了parseInt()和parseFloat()两个转换函数。前者把值转换成整数，后者把值转换成浮点数。parseInt parseFloat只有对String类型调用这些方法 而Number还可以时其他类型

Number()的强制类型转换与parseInt()和parseFloat()方法的处理方式相似，只是它转换的是整个值，而不是部分值Number( "5.6.7 ") //NAN parseFloat("5.6.7 ")//5.6

### 数据类型

基本数据类型：Number，String，Boolean，Undefined，Null

复杂数据类型：Object，Array，Function，RegExp，Date，Error

全局数据类型：Math

对于js的常见内置对象：Date, Array, Math, Number, Boolean, String, Array, RegExp, Function

### 全局函数

serTimeout是window的一个方法，如果把window当做全局对象来看待的话，它就是全局函数。严格来讲，它不是。全局函数与内置对象的属性或方法不是一个概念。全局函数它不属于任何一个内置对象。JavaScript 中包含以下 7 个全局函数escape( )、eval( )、isFinite( )、isNaN( )、parseFloat( )、parseInt( )、unescape( )----ECMAScript v3 反对使用该方法，应用使用 decodeURI() 和 decodeURIComponent() 替代它。

# 补码、反码、源码

正数：3个都一样

反码：除符号位外，正码按位取反

补码：反码+1, 高位溢出舍去

补码再求一次补码则得到原码

计算机一般以补码存

负数间转换。如-(1),-(-1)等。

1. 符号位取反；

2. 真值域取反；

3. 整体+1。

# Doctype 严格模式与混杂模式的区别在哪里?

1）、<!DOCTYPE> 声明位于文档中的最前面，处于 <html> 标签之前。告知浏览器以何种模式来渲染文档。

2）、严格模式的排版和 JS 运作模式是 以该浏览器支持的最高标准运行。

3）、在混杂模式中，页面以宽松的向后兼容的方式显示。模拟老式浏览器的行为以防止站点无法工作。

4）、DOCTYPE不存在或格式不正确会导致文档以混杂模式呈现。

# null和undefined的区别，null == undefined的结果是？

**null是一个表示”无”的对象，转为数值时为0；undefined是一个表示”无”的原始值，转为数值时为NaN。**

当声明的变量还未被初始化时，变量的默认值为undefined。

null用来表示尚未存在的对象，常用来表示函数企图返回一个不存在的对象。

undefined表示”缺少值”，就是此处应该有一个值，但是还没有定义。典型用法是：

（1）变量被声明了，但没有赋值时，就等于undefined。

（2) 调用函数时，应该提供的参数没有提供，该参数等于undefined。

（3）对象没有赋值的属性，该属性的值为undefined。

（4）函数没有返回值时，默认返回undefined。

null表示”没有对象”，即该处不应该有值。典型用法是：

（1） 作为函数的参数，表示该函数的参数不是对象。

（2） 作为对象原型链的终点。

# sass-css预处理器

缺点：简单来说CSS预处理器语言较CSS玩法变得更高级了，但同时降低了自己对最终代码的控制力。更致命的是提高了门槛，首先是上手门槛，其次是维护门槛，再来是团队整体水平和规范的门槛。这也造成了初学学习成本的昂贵。

优点：用一种专门的编程语言，为CSS增加了一些编程的特性，将CSS作为目标生成文件，然后开发者就只要使用这种语言进行编码工作。通俗的说，CSS预处理器用一种专门的编程语言，进行Web页面样式设计，然后再编译成正常的CSS文件，以供项目使用。CSS预处理器为CSS增加一些编程的特性，无需考虑浏览器的兼容性问题，例如你可以在CSS中使用变量、简单的逻辑程序、函数等等在编程语言中的一些基本特性，可以让你的CSS更加简洁、适应性更强、可读性更佳，更易于代码的维护等诸多好处。

# SVG与canva区别

SVG 是一种使用 XML 描述 2D 图形的语言。 SVG 基于 XML，这意味着 SVG DOM 中的每个元素都是可用的。您可以为某个元素附加 JavaScript 事件处理器。 在 SVG 中，每个被绘制的图形均被视为对象。如果 SVG 对象的属性发生变化，那么浏览器能够自动重现图形。 Canvas Canvas 通过 JavaScript 来绘制 2D 图形。 Canvas 是逐像素进行渲染的。 在 canvas 中，一旦图形被绘制完成，它就不会继续得到浏览器的关注。如果其位置发生变化，那么整个场景也需要重新绘制，包括任何或许已被图形覆盖的对象。 Canvas 与 SVG 的比较 下表列出了 canvas 与 SVG 之间的一些不同之处。 Canvas 依赖分辨率 不支持事件处理器 弱的文本渲染能力 能够以 .png 或 .jpg 格式保存结果图像 最适合图像密集型的游戏，其中的许多对象会被频繁重绘 SVG 不依赖分辨率 支持事件处理器 最适合带有大型渲染区域的应用程序（比如谷歌地图） 复杂度高会减慢渲染速度（任何过度使用 DOM 的应用都不快） 不适合游戏应用

# ECMAScript与JavaScript的规范

ECMAScript是JavaScript的规范，JavaScript是ECMAScript的实现。s是一种专为与网页交互而设计的脚本语言，由下列三个不同部分组成：

核心ECMAScript，提供核心语言功能；

文档对象模型（DOM），提供访问和操作网页内容的方法和接口；

浏览器对象模型（BOM），提供与浏览器交互的方法和接口。

JavaScript包括了核心ECMAScript。可以理解为：ECMAScript是js的一部分。

# 会造成内存泄漏的操作：

1、闭包引起的内存泄漏；

2、意外的全局变量引起的内存泄漏；

3、没有清理的DOM元素引起的内存泄漏；

4、被遗忘的定时器或者回调函数；

5、子元素存在引用引起的内存泄漏；

# CDN

# 跨域

1. JSONP
2. CORS
3. 通过修改document.domain来跨子域
4. 使用window.name来进行跨域
5. 使用HTML5中新引进的 window.postMessage 方法来跨域传送数据

# 什么是回调

# 什么是同步/异步

# 什么是I/O

# 什么是单线程/多线程

# 什么是阻塞/非阻塞

# 什么是事件

# 什么是事件驱动

# 什么是基于事件驱动的回调

# 什么是事件循环

# 浏览器输入网址后经历了什么

1 输入网址

2 域名解析

2.1 浏览器自身缓存（chrome://net-internals/#dns）

2.2 操作系统自身的dns缓存

2.3 读取本地的host文件

2.4 浏览器发起一个DNS的系统调用（本地主控DNS服务器：宽带运营商服务器查看本身缓存）

2.5 运营商服务器发起一个迭代DNS解析的请求（->根域.com服务器->imooc.com域服务器->运营商服

务器得到(结果)->操作系统内核并缓存->浏览器）

3 浏览器与服务器的web程序tcp链接->三次握手

4 发送http请求

5 服务器处理请求，并将结果返回

6 浏览器根据返回结果进行处理或页面渲染

# JS数组

## 非ES6

### 类数组的转化

Array.prototype.slice.call(arguments).reduce(function(x, y) {

return x + y;

### 数组的复制,去首尾可以用slice

[].Slice(0) [].slice(1) [].slice(0,-1)

### 数组求和

[].reduce(function(prev,next){}[,first][,second])

数组的遍历

Every some filter //都是返回新数组

Map(function(value,index){}) forEach//map有return foreach无return 都是返回新数组

Sort splice 这两个会改变原数组

### Num还可以这么转字符串啊

num.toString(2)//2进制

parseFloat(a\*b).toFixed(len)

### 巧用位运算

获取数字 num 二进制形式第 bit 位的值。注意：  
1、bit 从 1 开始  
2、返回 0 或 1  
3、举例：2 的二进制为 10，第 1 位为 0，第 2 位为 1

function valueAtBit(num, bit) {

return (num >> (bit -1)) & 1;

}

正则

Match与捕获

var str="1 plus 2 equal 3 4"

document.write(str.match(/(\d+)/g))//[1,2,3,4]

document.write(str.match((\d+/g))//[1,2,3,4]

document.write(str.match(/(\d+)/))//[1,1]

document.write(str.match(/(\d+)/))//1

 var a=b=3;这只是声明了a 没有声明b 顺序是b=3,var a=b;

contextmenu 是当浏览者按下鼠标右键出现菜单时或者通过键盘的按键触发页面菜单时触发的事件 [ 试试在页面中的 <body> 中加入 onContentMenu="return false" 就可禁止使用鼠标右键了 ]

# XML

1. <? Xml version=”1.0” encoding=“utf-8”?> 版本声明，可以省略，必须放第一行
2. 在 XML 中编写注释的语法与 HTML 的语法很相似。
3. HTML 会把多个连续的空格字符裁减（合并）为一个：
4. 在 Windows 应用程序中，换行通常以一对字符来存储：回车符（CR）和换行符（LF）。
5. 区分大小写，XML 元素必须被正确的嵌套
6. 无预定义标签，标签可以有**属性，**属性通常提供不属于数据组成部分的信息，必须加引号，应该尽量避免使用属性
7. 标签可使用任何名，没有保留的字词。XML 文档必须有一个根元素

# 逻辑分辨率和物理分辨率有什么区别 怎样通过js转化

# http2

可以设置HTTP请求的**时限**。

可以使用**FormData**对象管理表单数据。

可以上传**文件**。

可以请求不同域名下的数据（**跨域**请求CORS）。

可以获取服务器端的**二进制数据。(**responseType = 'blob'或是"arraybuffer"**)**

可以获得数据传输的**进度信息**。（xhr.onprogress，event.total，event.loaded是已经传输的字节。如果event.lengthComputable不为真，则event.total等于0。）

# JSONP

利用在页面中创建<script>节点的方法向不同域提交HTTP请求的方法称为JSONP，script标签的src属性是没有跨域的限制的。这样说来，这种跨域方式其实与ajax XmlHttpRequest协议无关了。这项技术可以解决跨域提交Ajax请求的问题。

JSONP的优点是：它不像XMLHttpRequest对象实现的Ajax请求那样受到同源策略的限制；它的兼容性更好，在更加古老的浏览器中都可以运行，不需要XMLHttpRequest或ActiveX的支持；并且在请求完毕后可以通过调用callback的方式回传结果。

JSONP的缺点则是：它只支持GET请求而不支持POST等其它类型的HTTP请求；它只支持跨域HTTP请求这种情况，不能解决不同域的两个页面之间如何进行JavaScript调用的问题。