HTML/CSS 部分

1、对 WEB 标准以及 W3C 的理解与认识

标签闭合、标签小写、不乱嵌套、提高搜索机器人搜索几率、使用外 链 css 和 js 脚本、结构行为表现的分离、文件下载与页面速度更快、 内容能被更多的用户所访问、内容能被更广泛的设备所访问、更少的代码和组件，容易维护、改版方便，不需要变动页面内容、提供打印版本 而不需要复制内容、提高网站易用性；

2、前端页面有哪三层构成，分别是什么?作用是什么

结构层 Html；主要指 DOM 节点；
样式（表示）层 CSS；主要是指页面渲染；
脚本（行为）层 Js：主要指页面动画效果；

3、html 和 xhtml 的区别

HTML 是一种基本的 WEB 网页设计语言，XHTML 是一个基于 XML 的置标语言 最主要的不同，
XHTML 元素：
必须被正确地嵌套；
所有的标记都必须要有一个相应的结束标记；
所有标签的元素和属性 的名字都必须使用小写；
XHTML 文档必须拥有根元素；
所有的属性必须用引号 "" 括起来；
把所有 < 和 & 特殊符号用编码表示；
给所有属性赋一个值；不要在注释内容中使用 "--"；
图片必须有说明文字。

4、Doctype

1） Doctype 作用： 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。DOCTYPE 不存在或格式不正确会导致文档以混杂模式呈现。 2）严格模式与混杂模式的区别 严格模式是浏览器根据 web 标准去解析页面，是一种要求严格的 DTD，不允许使用任何表现层的语法，如
，严格模式的排版是以该浏览器支持的最高标准运行，呈现遵循最新标准的网页； 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，用于呈现为传统浏览器而设计的网页。 3）如何触发 根据不同的 DTD 触发，如果没有声明，那么默认为混杂模式。 4）Doctype 文档类型 可声明三种 DTD 类型，分别表示严格版本、过渡版本以及基于框架的 HTML 文档。

5、行内元素、块级元素

块级元素： div ， p ， form, ul ， li ， ol, dl, dt,dd, form, address, fieldset, hr, menu, table, h1-h6、blockquote 各占一行，垂直方向排列。从新行开始结束接着一个断行。
行内元素: a b br，i，span, strong, em, img ， input, label, select, textarea, cite, button 可水平方向排列，不能包含块级元素，设置 width、height 无效，margin、padding 上下无效。 空元素： br、meta、hr、link、input、img

6、语义化

用正确的标签做正确的事情。
1. 直观的认识标签，去掉或者丢失样式的时候能够让页面呈现出清晰的结构
2. 有利于 SEO：和搜索引擎建立良好沟通，有助于爬虫抓取更多的有效信息：爬虫依赖于标签来确定上下文和各个关键字的权重；
3. 方便其他设备解析（如屏幕阅读器、盲人阅读器、移动设备）以意义的方式来渲染网页；
4. 便于团队开发和维护，语义化更具可读性，是下一步吧网页的重要动向，遵循 W3C 标准的团队都遵循这个标准，可以减少差异化

7、页面导入样式时，使用 link 和@import 有什么区别？

Link	@import
link 属于 XHTML 标签，除了加载 CSS 外，还能用于定义 RSS, 定义 rel 连接属性等作用，支持使用 javascript 改变样式；	@import 是 CSS 提供的，只能用于加载 CSS;
页面被加载时，link 会同时被加载	@import 引用的 CSS 会等到页面被加载完再加载
link 是 XHTML 标签，无兼容问题	@import 是 CSS2.1 提出的，只在 IE5 以上才能被识别
link 方式的样式的权重 高于@import 的权重.	@import 可以在 css 中再次引入其他样式表

8、href 与 src 的区别

Href——指向网络资源所在位置，建立和当前元素（锚点）或当前文档（链接）之间的链接，用于超链接。

Src——指向外部资源的位置，指向的内容将会嵌入到文档中当前标签所在位置；在请求 src 资源时会将其指向的资源下载并应用到文档内，例如 js 脚本，img 图片和 frame 等元素。当浏览器解析到该元素时，会暂停其他资源的下载和处理，直到将该资源加载、编译、执行完毕，图片和框架等元素也如此，类似于将所指向资源嵌入当前标签内。 这也是为什么将 js 脚本放在底部而不是头部。

9、标签上 title 与 alt 属性的区别是什么?

Alt 当图片不显示时用文字代表。 Title 为该属性提供信息

10、CSS 选择符

CSS 选择符有哪些?

id 选择器#；类选择器.；标签选择器 div, h1, p；相邻选择器 h1+p；子选择器 ul > li；后代选择器 li a；通配符选择器*；属性选择器 a[rel =“external”]）；伪类选择器 a: hover，li:nth-child。

继承:

可继承的样式： font-size; font-family; color; text-indent; UL;LI;DL;DD;DT;

不可继承的样式：border padding margin width height 优先级:

* 优先级就近原则，样式定义最近者为准; * 载入样式以最后载入的定位为准;

!important > 内联> id > class > tag|伪类|属性选择>伪对象>继承

11、Display position

display 的值的作用： 1.block 像块类型元素一样显示。

2.inline 缺省值，像行内元素类型一样显示。

3.inline-block 像行内元素一样显示，但其内容像块类型元素一样显示。

4.list-item 像块类型元素一样显示，并添加样式列表标记。

position 的值的定位区别： 1.absolute——绝对定位，脱离文档流定位。

2.fixed——绝对定位，脱离文档流定位，相对于浏览器窗口，fixed 元素与浏览器窗口之间的距离是不变的。

3.relative——相对定位，相对于其在普通流中的位置进行定位。

4.static——默认值，正常文档流

12、浮动

1.浮动元素脱离文档流，不占据空间。浮动元素碰到包含它的边框或者浮动元素的边框停留。

2.浮动元素引起的问题和解决办法

父元素的高度无法被撑开，影响与父元素同级的元素 与浮动元素同级的非浮动元素会跟随其后 若非第一个元素浮动，则该元素之前的元素也需要浮动，否则会影响页面显示的结构

3.清除浮动

1）空标签清除浮动：clear：both，但增加了无意义标签

2）给包含浮动元素的父标签添加 css 属性 overflow：auto; zoom:1。zoom:1 用于兼容 IE6。

3）after 伪对象清除浮动

```
#parent:after{
    content:".";
    height:0;
    visibility:hidden;
    display:block;
    clear:both;
}
```

13、居中

内联元素居中方案

水平居中设置：

1.行内元素： text-align:center;

2.Flex 布局：display:flex;justify-content:center;(支持 Chroime，Firefox，IE9+);

垂直居中设置：

1.父元素高度确定的单行文本（内联元素）：height = line-height;

2.父元素高度确定的多行文本（内联元素）：

a:父元素设置 display:table;

b:子元素设置 display:table-cell ， vertical-align:middle;

块级元素居中方案

```
1）relative居中
    .div{
        position: relative;
        margin: x auto;
    }

2）absolute居中
    a)  .div{
            position: absolute;
            left: 50%;
            top: 50%;
            margin-left: -width/2;
            margin-top: -height/2;
        }
    b)  .div{
            position: absolute;
            top: 0;
            right: 0;
            bottom: 0;
            left: 0;
            margin: auto;
        }
```

3）未知宽度高度块级元素居中 table-cell 可用于多行元素 vertical-align:middle;

```
.div4{
    display: table;
    width: 300px;
    height: 300px;
    background: #ff0;
    text-align: center;
}
.div4 span{
    display: table-cell;
    color: #00f;
    vertical-align: middle;
}
```

4）背景图片居中：background-position:center center;

14、浏览器内核，css 前缀

IE: trident 内核 -ms
Firefox： gecko 内核 -moz
Safari： webkit 内核 -webkit
Opera： 以前是 presto 内核，Opera 现已改用 Google Chrome 的 Blink 内核 -o
Chrome： webkit 内核 -ms
渲染引擎：负责取得网页的内容（HTML、XML、图像等等）、整理讯息（例如加入 CSS 等），以及计算网页的显示方式，然后会输出至显示器或打印机。浏览器的内核的不同对于网页的语法解释会有不同，所以渲染的效果也不相同。所有网页浏览器、电子邮件客户端以及其它需要编辑、显示网络内容的应用程序都需要内核。
JS 引擎：解析和执行 javascript 来实现网页的动态效果。
最开始渲染引擎和 JS 引擎并没有区分的很明确，后来 JS 引擎越来越独立，内核就倾向于只指渲染引擎。

15、HTML5 新特性

HTML5 现在已经不是 SGML 的子集，主要是关于图像，位置，存储，多任务等功能的增加。
(1)绘画 canvas;
(2)用于媒介回放的 video 和 audio 元素;
(3)本地离线存储 localStorage 长期存储数据，浏览器关闭后数据不丢失;
(4)sessionStorage 的数据在浏览器关闭后自动删除;
(5)语意化更好的内容元素，比如 article、footer、header、nav、section;
(6)表单控件，calendar、date、time、email、url、search;
(7)新的技术 webworker, websocket, Geolocation;
* ie9 以下不支持 html 新特性：直接使用成熟的框架、使用最多的是 html5shim 框架

```
<!--[if lt IE 9]>
<script> src="http://html5shim.googlecode.com/svn/trunk/html5.js"</script>
<![endif]-->
```

16、CSS3 新特性

1.圆角：border-radius，阴影：box-shadow，border-image
2. 对文字加特效（text-shadow），线性渐变（gradient），旋转（transform）
3. transform:rotate(9deg) scale(0.85,0.90) translate(0px,-30px) skew(-9deg,0deg);// 旋转,缩放,定位,倾斜
4. 增加了更多的 CSS 选择器 多背景 rgba

5. 媒体查询，多栏布局
6.新增伪类：
:root :root 选择文档的根元素。
:empty p:empty 选择没有子元素的每个 p 元素（包括文本节点）。
:target #news:target 选择当前活动的 #news 元素。
:enabled input:enabled 选择每个启用的 <input>元素。
:disabled input:disabled 选择每个禁用的 input 元素
:checked input:checked 选择每个被选中的 input 元素。
:not(selector) :not(p) 选择非 p 元素的每个元素。
::selection ::selection 选择被用户选取的元素部分。
:first-of-type p:first-of-type 选择属于其父元素的首个 p 元素的每个 p 元素。
:last-of-type p:last-of-type 选择属于其父元素的最后 p 元素的每个 p 元素。
:only-of-type p:only-of-type 选择属于其父元素唯一的 p 元素的每个 p 元素。
:only-child p:only-child 选择属于其父元素的唯一子元素的每个 p 元素。
:nth-child(n) p:nth-child(2) 选择属于其父元素的第二个子元素的每个 p 元素。
:nth-last-child(n) p:nth-last-child(2) 同上，从最后一个子元素开始计数。
:nth-of-type(n) p:nth-of-type(2) 选择属于其父元素第二个 p 元素的每个 p 元素。
:nth-last-of-type(n) p:nth-last-of-type(2) 同上，但是从最后一个子元素开始计数。
:last-child p:last-child 选择属于其父元素最后一个子元素每个 p 元素。

17、Bootstrap 了解程度

Bootstrap 是基于 HTML、CSS、JAVASCRIPT 的前端框架，它简洁灵活，使得 Web 开发更加快捷。 它由 Twitter 的设计师 Mark Otto 和 Jacob Thornton 合作开发，是一个 CSS/HTML 框架。

Bootstrap 提供了优雅的 HTML 和 CSS 规范，它即是由动态 CSS 语言 Less 写成。

Bootstrap 一经推出后颇受欢迎，一直是 GitHub 上的热门开源项目，包括 NASA 的 MSNBC（微软全国广播公司）的 Breaking News 都使用了该项目。

国内一些移动开发者较为熟悉的框架，如 WeX5 前端开源框架等，也是基于 Bootstrap 源码进行性能优化而来。

Bootstrap 是基于 HTML5 和 CSS3 开发的，它在 jQuery 的基础上进行了更为个性化和人性化的完善，形成一套自己独有的网站风格，并兼容大部分 jQuery 插件。

bootstrap 给我们提供了一系列的组件，排版、代码、按钮、图片、表格、表单、标题、导航条、分页、标签等等，让我们可以不用写大量代码，优化我们时间分配。

18、CSS Hack——针对不同的浏览器写不同的 CSS

```
IE6 BUG的解决方法
1.双边距BUG:
在IE6下，如果对元素设置了浮动，同时又设置了margin-left或margin-right，margin值会加倍。
box{ float:left; width:10px; margin:0 0 0 10px;}
这种情况之下IE会产生20px的距离
解决方案：在float的标签样式控制中加入 _display:inline; 将其转化为行内属性。( _ 这个符号只有ie6会识别)<br>

2.3像素问题 ：使用float引起的 ：dislpay:inline -3px

3. 超链接访问过后hover样式就不出现了 被点击访问过的超链接样式不在具有hover和active了解决方法是改变CSS属性的排列顺序:
    L-V-H-A :  a:link {} a:visited {} a:hover {} a:active {}

4.Ie z-index问题：给父级添加position:relative
5.Png 透明 使用js代码 改
png24位的图片在iE6浏览器上出现背景，解决方案是做成PNG8
6.Min-height 最小高度 ！Important 解决’
7.select 在ie6下遮盖 使用iframe嵌套
8.为什么没有办法定义1px左右的宽度容器（IE6默认的行高造成的，使用over:hidden,zoom:0.08 line-height:1px）
9、渐进识别的方式，从总体中逐渐排除局部。
首先，巧妙的使用“\9”这一标记，将IE游览器从所有情况中分离出来。
接着，再次使用 "+" 将IE8和IE7、IE6分离开来，这样IE8已经独立识别。
.bb{
    background-color:#f1ee18; /*所有识别*/
    .background-color:#00deff\9; /*IE6、7、8识别*/
    +background-color:#a200ff; /*IE6、7识别*/
    _background-color:#1e0bd1; /*IE6识别*/
}<br>

    *  IE下,可以使用获取常规属性的方法来获取自定义属性,
       也可以使用getAttribute()获取自定义属性;
       Firefox下,只能使用getAttribute()获取自定义属性.
       解决方法:统一通过getAttribute()获取自定义属性.

    *  IE下,even对象有x,y属性,但是没有pageX,pageY属性;
      Firefox下,event对象有pageX,pageY属性,但是没有x,y属性.

    *  （条件注释）缺点是在IE浏览器下可能会增加额外的HTTP请求数。
```

19、描述 css reset 的作用和用途

因为浏览器的兼容问题，不同浏览器对有些标签的默认值是不同的，如果没对 CSS 初始化往往会出现浏览器之间的页面显示差异。Reset 重置浏览器的 css 默认属性，让他们统一。

当然，初始化样式会对 SEO 有一定的影响，但鱼和熊掌不可兼得，但力求影响最小的情况下初始化。

淘宝的样式初始化：

```
body, h1, h2, h3, h4, h5, h6, hr, p, blockquote, dl, dt, dd, ul, ol, li, pre, form, fieldset,
legend, button, input, textarea, th, td { margin:0; padding:0; }
body, button, input, select, textarea { font:12px/1.5tahoma, arial, \5b8b\4f53; }
h1, h2, h3, h4, h5, h6{ font-size:100%; }
address, cite, dfn, em, var { font-style:normal; }
code, kbd, pre, samp { font-family:couriernew, courier, monospace; }
small{ font-size:12px; }
ul, ol { list-style:none; }
a { text-decoration:none; }
a:hover { text-decoration:underline; }
sup { vertical-align:text-top; }
sub{ vertical-align:text-bottom; }
legend { color:#000; }
fieldset, img { border:0; }
button, input, select, textarea { font-size:100%; }
table { border-collapse:collapse; border-spacing:0; }
```

20、解释 css sprites，如何使用。

Css 精灵 把一堆小的图片整合到一张大的图片上，减轻服务器对图片的请求数量

21、CSS 盒子模型

元素的内容（content），内边距（padding），边框（border），边距（margin）四个部分一起构成 css 中元素的盒模型。

标准 W3C 盒模型：一个块的总宽度= width + margin(左右) + padding(左右) + border(左右)；
先做鞋，后做盒 多用于 pc content 部分不包含其他部分
怪异（IE）盒模型：一个块的总宽度= width + margin(左右)（即 width 已经包含了 padding 和 border 值）；
先做盒，后做鞋 多用于移动端 box-sizing: border-box

22、css 的基本语句构成 1
选择器{属性 1:值 1;属性 2:值 2;……}

23、CSS 引入的方式有哪些?
内联 内嵌 外链 导入

24、px 和 em 的区别
px 和 em 都是长度单位，区别是，px 的值是固定的，指定是多少就是多少，计算比较容易。em 得值不是固定的，并且 em 会继承父级元素的字体大小。 浏览器的默认字体高都是 16px。所以未经调整的浏览器都符合: 1em=16px。那么 12px=0.75em, 10px=0.625em Px：值固定 容易计算 Em：值不固定 会继承父级元素的字体大小

25、iframe 有那些缺点？
*iframe 会阻塞主页面的 Onload 事件； *iframe 和主页面共享连接池，而浏览器对相同域的连接有限制，所以会影响页面的并行加载。 使用 iframe 之前需要考虑这两个缺点。如果需要使用 iframe，最好是通过 javascript 动态给 iframe 添加 src 属性值，这样可以可以绕开以上两个问题。

JavaScript 部分

一、数据类型：
基本数据类型：String,boolean,Number,Undefined, Null
引用数据类型：Object(Array,Date,RegExp,Function) （复合数据类型）
1、两者的区别： 1）值存储方式不同： 原始数据类型：将变量名和值都存储在栈内存中
引用数据类型：将变量名存储在栈内存中，将值存储在堆内存中，并在栈内存中存储值的地址，该地址指向堆内存中的值。
2）赋值方式不同： 给一个变量赋予另一个变量的值
若值为原始数据类型，直接在栈内存中生成值，两个变量以后进行值改变不会相互影响
若值为引用数据类型，赋予变量的是值地址，通过这个地址，两者指向的其实是堆内存中的同一个值，某个变量对值进行改变，会直接影响另一个变量的值

2. 解释清楚 null 和 undefined
Null：赋值为“空值”,从逻辑角度来看，null 值表示空对象指针；
undefined 声明未赋值。

3.数据类型转换 转换函数： parseInt() parseFloat()
只有对 String 类型调用这两个函数才能正确运行；对其他类型返回的都是 NaN；
parseInt 和 parseFloat 只转换第一个无效字符之前的字符串，如果遇到非法字符，会忽略之后的所有内容。
（NaN——JavaScript 中唯一一个不等于自己的值。(NaN == NaN) === false）！
强制数据类型转换：
Boolean(value)——把给定的值转换成 Boolean 型；
Number(value)——把给定的值转换成数字（可以是整数或浮点数）；
String(value)——把给定的值转换成字符串。
隐式类型转换（”==”和“===”的不同：前者会自动转换类型，后者不会）
b1）四则运算：
加法运算符+：只要其中一个是 String 类型，表达式的值便是一个 String。

其他的四则运算，只要其中一个是 Number 类型，表达式的值便是一个 Number。
对于非法字符的情况通常会返回 NaN：
2）判断语句：中的判断条件需要是 Boolean 类型，所以条件表达式会被隐式转换为 Boolean
3）Alert：会根据对象类型，隐式转换转换为 string 或 number

4. 类型识别的方法：
typeof: 例子：（typeof 1 返回结果："number" typeof new Array()
返回结果:"object"）可识别标准类型，除了 null 之外。不能识别具体的对象类型，除了 function 之外。

instanceof：例子：（[] instanceof Array 返回结果：true 'a' instanceof String
返回结果：false）可以判别内置对象类型，不能判断原始类型值

constructor：例子（'123'.constructor == String 返回结果：true）

Object.prototype.toString.call：例子(Object.prototype.toString.call(1) 返回结果:"[object Number]")
可以识别标准数据类型和内置对象类型。不能识别自定义对象类型

5.javascript 的常用对象有哪些？
String, Math, Date 和 Array 对象

6. JavaScript 的数据对象有那些属性值？
js 的常见内置对象类：Date,Array,Math、Number、Boolean、String、Array、RegExp、Function...
writable：这个属性的值是否可以改。
configurable：这个属性的配置是否可以删除，修改。
enumerable：这个属性是否能在 for…in 循环中遍历出来或在 Object.keys 中列举出来。
value：属性值。
* 当我们需要一个属性的时，Javascript 引擎会先看当前对象中是否有这个属性， 如果没有的话，就会查找他的 Prototype 对象是否有这个属性。
1）创建一个对象

```
function Person(name, age) {
  this.name = name;
  this.age = age;
  this.sing = function() { alert(this.name) }
}
```

2）谈谈 This 对象的理解。
this 是 js 的一个关键字，随着函数使用场合不同，this 的值会发生变化。 但是总有一个原则，那就是 this 指的是调用函数的那个对象。
this 一般情况下：是全局对象 Global。 作为方法调用，那么 this 就是指这个对象 3）如何判断一个对象是否属于某个类？
使用 instanceof

```
if(a instanceof Person){
   alert('yes');
}
```

二、javascript 是面向对象的，怎么体现 javascript 的继承关系？

使用 prototype 来实现。

1. 如何理解 JavaScript 原型链

* 原型链是由一些用来继承和共享属性的对象组成的（有限的）对象链。 JavaScript 中的每个对象都有一个 prototype 属性，我们称之为原型，而原型的值也是一个对象，因此它也有自己的原型，这样就串联起来了一条原型链，原型链的链头是 object,它的 prototype 比较特殊，值为 null。

原型链的作用是用于对象继承，函数 A 的原型属性(prototype property)是一个对象，当这个函数被用作构造函数来创建实例时，该函数的原型属性将被作为原型赋值给所有对象实例，比如我们新建一个数组，数组的方法便从数组的原型上继承而来。 当访问对象的一个属性时，首先查找对象本身，找到则返回；若未找到，则继续查找其原型对象的属性(如果还找不到实际上还会沿着原型链向上查找，直至到根)．只要没有被覆盖的话，对象原型的属性就能在所有的实例中找到，若整个原型链未找到则返回 undefined；

2．b 继承 a 的方法

```
解决方案1:
function A(name){
    this.name = name;
    this.sayHello = function(){alert(this.name+" say Hello!");};
}

function B(name,id){
    this.temp = A;
    this.temp(name);          //相当于new A();
    delete this.temp;
    this.id = id;
    this.checkId = function(ID){alert(this.id==ID)};
}
解决方案2:
b.prototype=new a;

3.new构建对象的本质
function User(){
 //this = {};
 //this.constructor = User;
 this.name = "Vicfeel";
 this.age = 23;
 //return this;
}
var user = new User();
通过new操作符，实际上在构造函数User中完成了如下操作:
  1、创建一个空对象，并且 this 变量引用该对象，同时还继承了该函数的原型。
  2、执行构造函数，属性和方法被加入到 this 引用的对象中。
  3、新创建的对象由 this 所引用，并且最后隐式的返回 this（新创建的对象） ;
  构造函数默认返回的新创建的this对象，如果手动return 一个变量的话，如果该变
```

三、怎样添加、移除、移动、复制、创建和查找节点

（1）创建新节点

createDocumentFragment() //创建一个 DOM 片段

createElement () //创建一个具体的元素

createTextNode() //创建一个文本节点

（2）添加、移除、替换、插入

append ()

removeChild()

replaceChild()

insertBefore()

（3）查找

getElementsByTagName_r() //通过标签名称

getElementsByName() //通过元素的 Name 属性的值

getElementByClassName() //通过元素 class

getElementById() //通过元素 Id，唯一性

四、如何理解和应用 JavaScript 闭包

全局变量：当前页面内有效；局部变量：方法内有效；函数内部声明变量的时候，一定要使用 var 命令。如果不用的话，你实际上声明了一个全局变量！

1. 闭包就是能够读取其他函数内部变量的函数。 由于在 Javascript 语言中，只有函数内部的子函数才能读取局部变量，因此可以把闭包简单理解成"定义在一个函数内部的函数"。子函数能被外部调用到，则该作用连上的所有变量都会被保存下来，所以，在本质上，闭包就是将函数内部和函数外部连接起来的一座桥梁。

```
function f1(){
    var n=999;
    nAdd=function(){n+=1}
    function f2(){
        alert(n);
    }
    return f2;
}
var result=f1();
result(); // 999
nAdd();
result(); // 1000
```

以上，result 即为闭包，运行了两次，第一次的值是 999，第二次的值是 1000。这证明了，函数 f1 中的局部变量 n 一直保存在内存中，并没有在 f1 调用后被自动清除。

2.闭包用途：

闭包最大用处有两个，一个是可以读取函数内部的变量，另一个就是让这些变量的值始终保持在内存中。

3、使用闭包的注意点

1）由于闭包会使得函数中的变量都被保存在内存中，内存消耗很大，所以不能滥用闭包，否则会造成网页的性能问题，在 IE 中可能导致内存泄露。解决方法：在退出函数之前，将不使用的局部变量全部删除。

2）闭包会在父函数外部，改变父函数内部变量的值。所以，如果你把父函数当作对象（object）使用，把闭包当作它的公用方法（Public Method），把内部变量当作它的私有属性（private value），这时一定要小心，不要随便改变父函数内部变量的值。

五、谈一谈 JavaScript 作用域链

当执行一段 JavaScript 代码（全局代码或函数）时，JavaScript 引擎会创建为其创建一个作用域又称为执行上下文（Execution Context），在页面加载后会首先创建一个全局的作用域，然后每执行一个函数，会建立一个对应的作用域，从而形成了一条作用域链。每个作用域都有一条对应的作用域链，链头是全局作用域，链尾是当前函数作用域。 作用域链的作用是用于解析标识符，当函数被创建时（不是执行），会将 this、arguments、命名参数和该函数中的所有局部变量添加到该当前作用域中，当 JavaScript 需要查找变量 X 的时候（这个过程称为变量解析），它首先会从作用域链中的链尾也就是当前作用域进行查找是否有 X 属性，如果没有找到就顺着作用域链继续查找，直到查找到链头，也就是全局作用域链，仍未找到该变量的话，就认为这段代码的作用域链上不存在 x 变量，并抛出一个引用错误（ReferenceError）的异常。

六、JavaScript 变量声明提前

JavaScript 变量在声明之前已经可用，JavaScript 的这个特性被非正式的称为声明提前（hoisting），即 JavaScript 函数中声明的所有变量（但不涉及赋值）都被“提前”至函数的顶部。 var a = “” ; 不是 not found 而是 undefined

七、.浏览器的对象模型？

window 顶级对象

window.alert(msg) window.prompt() window.confirm() if(window.confirm()){ ... } window.open()

window.close() document document.write()

history: 当用户浏览网页时，浏览器保存了一个最近所访问网页的 url 列表。这个列表就是用 history 对象表示。

history.back():后退 history.forward():前进 history.go(n):正数表示向前，负数表示向后

location: 表示当前打开的窗口或框架的 URL 信息。

location.href：重定向，等价于 location.assign(url)； location.host：类似 www.163.com:80

navigator: 表示浏览器的信息及 js 运行的环境

navigator.cookieEnabled：该属性表示是否启用 cookie

screen: 用于显示网页的显示器的大小和颜色

screen.width/screen.height：表示显示器的分辨率（总的宽度，高度）

八、Ajax

1、Ajax 是异步 JavaScript 和 XML，用于在 Web 页面中实现异步数据交互。

优点：

1)最大的一点是页面无刷新，用户的体验非常好。

2)使用异步方式与服务器通信，具有更加迅速的响应能力。

3)可以把以前一些服务器负担的工作转嫁到客户端，利用客户端闲置的能力来处理，减轻服务器和带宽的负担，节约空间和宽带租用成本。并且减轻服务器的负担，ajax 的原则是“按需取数据”，可以最大程度的减少冗余请求，和响应对服务器造成的负担，提升站点性能。

4)基于标准化的并被广泛支持的技术，不需要下载插件或者小程序，但需要客户允许 JavaScript 在浏览器上执行。

5)Ajax 使 WEB 中的界面与应用分离（也可以说是数据与呈现分离），有利于分工合作、减少非技术人员对页面的修改造成的 WEB 应用程序错误、提高效率、也更加适用于现在的发布系统。

缺点：

1)ajax 不支持浏览器 back 按钮，对搜索引擎不友好。

2)安全问题：AJAX 暴露了与服务器交互的细节。

3)对搜索引擎的支持比较弱：如果使用不当，可能造成请求数的增加，增大网络数据的流量，从而降低整个系统的性能。

4)破坏了程序的异常机制。

5)不容易调试。

6)违背 URL 和资源定位的初衷。

例如，我给你一个 URL 地址，如果采用了 Ajax 技术，也许你在该 URL 地址下面看到的和我在这个 URL 地址下看到的内容是不同的。这个和资源定位的初衷是相背离的。
7)AJAX 不能很好支持移动设备。
8)客户端过肥，太多客户端代码造成开发上的成本。
编写复杂、容易出错 ；冗余代码比较多（层层包含 js 文件是 AJAX 的通病，再加上以往的很多服务端代码现在放到了客户端）；破坏了 Web 的原有标准。
9)跨域问题： jsonp、 iframe、window.name、window.postMessage、服务器上设置代理页面
如何解决跨域问题：jquery-jsonp 插件

2、Ajax 工作原理
Ajax 的工作原理相当于在用户和服务器之间加了一个中间层(AJAX 引擎),使用户操作与服务器响应异步化。并不是所有的用户请求都提交给服务器,像一些数据验证和数据处理等都交给 Ajax 引擎自己来做, 只有确定需要从服务器读取新数据时再由 Ajax 引擎代为向服务器提交请求。
Ajax 其核心由 JavaScript、XMLHTTPRequest、DOM 对象组成，通过 XmlHttpRequest 对象来向服务器发异步请求，从服务器获得数据，然后用 JavaScript 来操作 DOM 而更新页面。这其中最关键的一步就是从服务器获得请求数据。
(1).XMLHTTPRequest 对象
Ajax 可无刷新更新页面,主要得益于 XMLHTTP 组件 XMLHTTPRequest 对象。
(2). Ajax 引擎，实际上是一个比较复杂的 JavaScript 应用程序，用来处理用户请求，读写服务器和更改 DOM 内容。 过程：
创建 ajax 对象（XMLHttpRequest/ActiveXObject(Microsoft.XMLHttp)）；
判断数据传输方式(GET/POST)；
打开链接 open()；
发送 send()；
当 ajax 对象完成第四步（onreadystatechange）数据接收完成，判断 http 响应状态（status）200-300 之间或者 304（缓存）执行回调函数

```
$.ajax({
    url: 'getnews.php',
    type: 'get',
    datatype: 'json',
    success: function(data) {

    }
    Error:function(){

    }
});
```

3.ajax 请求的时候 get 和 post 方式的区别
1）GET 请求会将参数跟在 URL 后进行传递，而 POST 请求则是作为 HTTP 消息的实体内容发送给 WEB 服务器。当然在 Ajax 请求中，这种区别对用户是不可见的
2）GET 方式请求的数据会被浏览器缓存起来，因此其他人就可以从浏览器的历史记录中读取到这些数据，例如账号和密码等。在某种情况下，GET 方式会带来严重的安全问题。而 POST 方式相对来说就可以避免这些问题。
get 请求和 post 请求在服务器端的区别:
3）在客户端使用 get 请求时,服务器端使用 Request.QueryString 来获取参数,而客户端使用 post 请求时,服务器端使用 Request.Form 来获取参数.
4）HTTP 标准包含这两种方法是为了达到不同的目的。POST 用于创建资源，资源的内容会被编入 HTTP 请示的内容中。例如，处理订货表单、在数据库中加入新数据行等。

当请求无副作用时（如进行搜索），便可使用 GET 方法；当请求有副作用时（如添加数据行），则用 POST 方法。一个比较实际的问题是：GET 方法可能会产生很长的 URL，或许会超过某些浏览器与服务器对 URL 长度的限制。
若符合下列任一情况，则用 POST 方法：
* 请求的结果有持续性的副作用，例如，数据库内添加新的数据行。
* 若使用 GET 方法，则表单上收集的数据可能让 URL 过长。
* 要传送的数据不是采用 7 位的 ASCII 编码。
若符合下列任一情况，则用 GET 方法：
* 请求是为了查找资源，HTML 表单数据仅用来帮助搜索。
* 请求结果无持续性的副作用。
* 收集的数据及 HTML 表单内的输入字段名称的总长不超过 1024 个字符。

4、Pjax：
pjax 是一种基于 ajax+history.pushState 的新技术，该技术可以无刷新改变页面的内容，并且可以改变页面的 URL。（关键点：可以实现 ajax 无法实现的后退功能）pjax 是 ajax+pushState 的封装，同时支持本地存储、动画等多种功能。目前支持 jquery、qwrap、kissy 等多种版本。

5、Json
JSON 是一种数据交换格式，而 JSONP 是一种依靠开发人员的聪明才智创造出的一种非官方跨域数据交互协议。一个是描述信息的格式，一个是信息传递双方约定的方法。 JSON 的优点：
1)基于纯文本，跨平台传递极其简单； {'age':'12', 'name':'back'}
2)Javascript 原生支持，后台语言几乎全部支持；
3)轻量级数据格式，是基于 JavaScript 的一个子集。数据格式简单, 易于读写, 占用字符数量极少，特别适合互联网传递；
4)可读性较强，虽然比不上 XML 那么一目了然，但在合理的依次缩进之后还是很容易识别的；
5)容易编写和解析，当然前提是你要知道数据结构；
JSON 的格式或者叫规则：
JSON 能够以非常简单的方式来描述数据结构，XML 能做的它都能做，因此在跨平台方面两者完全不分伯仲。
1)JSON 只有两种数据类型描述符，大括号{}和方括号[]，其余英文冒号:是映射符，英文逗号,是分隔符，英文双引号""是定义符。
2)大括号{}用来描述一组“不同类型的无序键值对集合”（每个键值对可以理解为 OOP 的属性描述），方括号[]用来描述一组“相同类型的有序数据集合”（可对应 OOP 的数组）。
3)述两种集合中若有多个子项，则通过英文逗号,进行分隔。
4)键值对以英文冒号:进行分隔，并且建议键名都加上英文双引号”″，以便于不同语言的解析。
5)JSON 内部常用数据类型无非就是字符串、数字、布尔、日期、null 这么几个，字符串必须用双引号引起来，其余的都不用，日期类型比较特殊，这里就不展开讲述了，只是建议如果客户端没有按日期排序功能需求的话，那么把日期时间直接作为字符串传递就好，可以省去很多麻烦。
Ajax 请求时，如何解析 json 数据：eval（）、parse（）
JSON.parse()可以解析 json 格式的数据，并且会对要解析的字符串进行格式检查，如果格式不正确则不进行解析，而 eval()则可以解析任何字符串，eval 是不安全的，鉴于安全性考虑 使用 parse 更靠谱。 如果用恶意用户在 json 字符串中注入了向页面插入木马链接的脚本，用 eval 也是可以操作的，而用 JSON.parse()则不必担心这个问题。
6、Jsonp：
(JSON with Padding)是一种跨域请求方式。主要原理是利用了 script 标签可以跨域请求的特点，由其 src 属性发送请求到服务器，服务器返回 js 代码，网页端接受响应，然后就直接执行了，这和通过 script 标签引用外部文件的原理是一样的。JSONP 由两部分组成：回调函数和数据，回调函数一般是由网页端控制，作为参数发往服务器端，服务器端把该函数和数据拼成字符串返回。该协议的一个要点就是允许用户传递一个 callback 参数给服务端，然后服务端返回数据时会将这个 callback 参数作为函数名来包裹住 JSON 数据，这样客户端就可以随意定制自己的函数来自动处理返回数据了。
7、Ajax Jsonp

1)ajax 和 jsonp 这两种技术在调用方式上“看起来”很像，目的也一样，都是请求一个 url，然后把服务器返回的数据进行处理，因此 jquery 和 ext 等框架都把 jsonp 作为 ajax 的一种形式进行了封装；

2)但 ajax 和 jsonp 其实本质上是不同的东西。ajax 的核心是通过 XmlHttpRequest 获取非本页内容，而 jsonp 的核心则是动态添加‘script’标签来调用服务器提供的 js 脚本。

3)所以说，其实 ajax 与 jsonp 的区别不在于是否跨域，ajax 通过服务端代理一样可以实现跨域，jsonp 本身也不排斥同域的数据的获取。

4)还有就是，jsonp 是一种方式或者说非强制性协议，如同 ajax 一样，它也不一定非要用 json 格式来传递数据，如果你愿意，字符串都行，只不过这样不利于用 jsonp 提供公开服务。

总而言之，jsonp 不是 ajax 的一个特例，哪怕 jquery 等巨头把 jsonp 封装进了 ajax，也不能改变这一点！

九、事件

1. Javascript 的事件流模型都有什么？

•事件冒泡（IE 事件流）：事件开始由最具体的元素接受，然后逐级向上传播

•事件捕捉（Netscape 事件流）：事件由最不具体的节点先接收，然后逐级向下，一直到最具体的

•DOM 事件流（先捕获后冒泡）：三个阶段：事件捕捉，目标阶段，事件冒泡

IE 和 DOM 事件流的区别：

1.执行顺序不一样；2.参数不一样；3.事件加不加 on （IE 加 on）；4.this 指向问题（IE 下指向 windows）

事件侦听函数的区别

```
IE使用：
[Object].attachEvent("name_of_event_handler", fnHandler); //绑定函数
[Object].detachEvent("name_of_event_handler", fnHandler); //移除绑定
DOM使用：
[Object].addEventListener("name_of_event", fnHandler, bCapture); //绑定函数
[Object].removeEventListener("name_of_event", fnHandler, bCapture); //移除
```

2.当一个 DOM 节点被点击时候，我们希望能够执行一个函数，应该怎么做？

- 直接在 DOM 里绑定事件：onclick=”test()”
- 在 JS 里通过 onclick 绑定：xxx.onclick = test
- 通过事件添加进行绑定：addEventListener(‘click’, test,true) True：捕获事件；False：冒泡事件

3.如何阻止事件冒泡和默认事件

阻止事件冒泡：e.stopPropagation(); window.event.cancelBubble=true;-IE;

阻止默认事件：e.preventDefault(); window.event.returnValue = false;IE;

4.普通事件和事件绑定有什么区别

```
普通事件：DOM0级事件—只支持单个事件，会被其他onclick事件覆盖<br>
btn.onclick = function () {
    alert('普通事件1');//不执行
}
btn.onclick = function () {
    alert('普通事件2');//弹出
}

事件绑定：DOM2级事件—可以添加多个事件而不用担心被覆盖
btn.addEventListener('click', function  () {
            alert('事件绑定1');//弹出
        },false);
        btn.addEventListener('click', function  () {
            alert('事件绑定2');//弹出
        },false);
上面代码执行后依次会弹出：普通事件2、事件绑定1、事件绑定2。
```

5.事件代理

让利用事件冒泡的原理，让自己的所触发的事件，让他的父元素代替执行！ 当我们需要对很多元素添加事件的时候，可以通过将事件添加到它们的父节点而将事件委托给父节点来触发处理函数。 比如我们需要向一个 ul 中动态添加很多个 li，需要遍历 li 逐个添加点击事件

```
var liNode = ulList.getElementByTagName("li");
for(var i=0, l = liNodes.length; i < l; i++){
liNode[i].onClick = function(){
    //li点击事件
}
```

众所周知，DOM 操作是十分消耗性能的。所以重复的事件绑定简直是性能杀手。而事件代理的核心思想，就是通过尽量少的绑定，去监听尽量多的事件。如何做呢？答案是利用事件冒泡机制,对其父节点 ul 进行事件绑定（Event Bubble),然后通过 event.target 来判断是哪个节点触发的事件，从而减少很多 EventHandler 的绑定。

```
var liNode = ulList.getElementByTagName("li");
liNode.onClick = function(e){
 if(e.target && e.target.nodeName.toUpperCase == "LI") {
  // li点击事件
 }
```

适合用事件委托的事件：click，mousedown，mouseup，keydown，keyup，keypress。

值得注意的是，mouseover 和 mouseout 虽然也有事件冒泡，但是处理它们的时候需要特别的注意，因为需要经常计算它们的位置，处理起来不太容易。

不适合的就有很多了，举个例子，mousemove，每次都要计算它的位置，非常不好把控，在不如说 focus，blur 之类的，本身就没用冒泡的特性，自然就不能用事件委托了。

6.js 是什么，js 和 html 的开发如何结合? js 是一种基于对象和事件驱动，并具有安全性的脚本语言。

可以 html 的三个地方编写 js 脚本语言：

一是在网页文件的标签对中直接编写脚本程序代码；

二是将脚本程序代码放置在一个单独的文件中，在网页文件中引用这个脚本程序语言；

三是将脚本程序代码作为某个元素的事件属性值或超链接的 href 属性值。

十、call 和 apply 的区别

```
obj.call(thisObj, arg1, arg2, ...);
obj.apply(thisObj, [arg1, arg2, ...]);
两者作用一致，都是把obj(即this)绑定到thisObj，这时候thisObj具备了obj的属性和方法。或者说thisObj『继承』了obj的属性和方法。
唯一区别是apply接受的是数组参数，call接受的是连续参数。
function add(j, k){
    return j+k;
}
function sub(j, k){
    return j-k;
}

add(5,3); //8
add.call(sub, 5, 3); //8
add.apply(sub, [5, 3]); //8

sub(5, 3); //2
sub.call(add, 5, 3); //2
sub.apply(add, [5, 3]); //2
通过call和apply，我们可以实现对象继承。
```

十一、IE 和标准下有哪些兼容性的写法

Var ev = ev || window.event

document.documentElement.clientWidth || document.body.clientWidth

Var target = ev.srcElement||ev.target

十二、js 中的 3 种弹出式消息提醒（警告窗口，确认窗口，信息输入窗口）的命令式什么？

alert 警告

confirm 确认

prompt 提问

十三、document load 和 document ready 的区别

Document.onload 页面包含图片等文件在内的所有元素都加载完成

Document.ready 文档结构加载完毕，不包含图片等非文字媒体文件，js 原生中没有，jquery 中有

$().ready(function)

十四、如何控制 alert 中的换行

```
function alert_br(){
  if(!document.all)//FF/{谷歌浏览器用这个
    alert('第一行\n第二行');
  }else{  //IE系列用这个
    alert('第一行\r\n第二行');
  }
}
alert_br();
```

十五、Node.js

1、Node.js 是一个基于 Chrome V8 引擎的 JavaScript 运行环境。

Node.js 使用了一个事件驱动、非阻塞式 I/O 的模型，使其轻量又高效。Node.js 的包管理器 npm，是全球最大的开源库生态系统。

Node.js 是一个后端的 Javascript 运行环境，意味着你可以编写系统级或者服务器端的 Javascript 代码，交给 Node.js 来解释执行，为 Javascript 提供了其他语言能够实现的许多功能。

*优点：

因为 Node 是基于事件驱动和无阻塞的，所以非常适合处理并发请求， 因此构建在 Node 上的代理服务器相比其他技术实现（如 Ruby）的服务器表现要好得多。 此外，与 Node 代理服务器交互的客户端代码是由 javascript 语言编写的， 因此客户端和服务器端都用同一种语言编写，这是非常美妙的事情。

*缺点：

Node 是一个相对新的开源项目，所以不太稳定，它总是一直在变， 而且缺少足够多的第三方库支持。看起来，就像是 Ruby/Rails 当年的样子。

2、Node.js 模块和包：

1）模块：

Node.js 官方提供了很多模块，这些模块分别实现了一种功能，如操作文件模块 fs,构建 http 服务模块的 http 等，每个模块都是一个 JS 文件，当然也可以自己编 写模块。

2）包

包可以将多个具有依赖关系的模块组织在一起，封装多个模块，以方便管理。Node.js 采用了 CommonJS 规范，根据 CommonJS 规范规定，一个 JS 文件就是 一个模块，而包是一个文件夹，包内必须包含一个 JSON 文件，命名 package.json。一般情况下，包内 bin 文件夹存放二进制文件，包内的 lib 文件夹存放 JS 文件，包内的 doc 文件夹存放文档，包内的 test 文件夹存放单元测试。package.json 文件中需要包含的字段及包的使用。

3） npm 包管理工具

npm 是 node.js 的包管理工具，npm 定义了包依赖关系标准，我们使用 npm 主要用来下载第三方包和管理本地下载的第三方包。

十六、js 延迟加载的几种方法

js 的延迟加载有助于提高页面的加载速度，以下是延迟加载的几种方法：

1.使用 setTimeout 延迟方法的加载时间，延迟加载 js 代码，给网页加载留出更多时间

```
function A(){
    $.post("/lord/login",{name:username,pwd:password},function(){
        alert("Hello");
    });
}
$(function (){
    setTimeout('A()', 1000); //延迟1秒
})
```

2.让 js 最后加载

例如引入外部 js 脚本文件时，如果放入 html 的 head 中,则页面加载前该 js 脚本就会被加载入页面，而放入 body 中，则会按照页面从上倒下的加载顺序来运行 JavaScript 的代码~~~ 所以我们可以把 js 外部引入的文件放到页面底部，来让 js 最后引入，从而加快页面加载速度

3.上述方法 2 也会偶尔让你收到 Google 页面速度测试工具的“延迟加载 javascript”警告。所以这里的解决方案将是来自 Google 帮助页面的推荐方案。 //这些代码应被放置在 body 标签前(接近 HTML 文件底部)

```
function downloadJSAtOnload() {
    var element = document.createElement("script");
    element.src = "defer.js";
    document.body.appendChild(element);
}
if (window.addEventListener)
    window.addEventListener("load", downloadJSAtOnload, false);
else if (window.attachEvent)
    window.attachEvent("onload", downloadJSAtOnload);
else window.onload = downloadJSAtOnload;
```

这段代码意思是等到整个文档加载完后，再加载外部文件“defer.js”。

使用此段代码的步骤：

1）.复制上面代码

2）.粘贴代码到 HTML 的标签前 (靠近 HTML 文件底部)

3）.修改“defer.js”为你的外部 JS 文件名

4）.确保你文件路径是正确的。例如：如果你仅输入“defer.js”，那么“defer.js”文件一定与 HTML 文件在同一文件夹下。

注意：这段代码直到文档加载完才会加载指定的外部 js 文件。因此，不应该把那些页面正常加载需要依赖的 javascript 代码放在这里。而应该将 JavaScript 代码分成两组。一组是因页面需要而立即加载的 javascript 代码，另外一组是在页面加载后进行操作的 javascript 代码(例如添加 click 事件或其他东西)。这些需等到页面加载后再执行的 JavaScript 代码，应放在一个外部文件，然后再引进来。

异步加载的方式：defer 和 async

(1) defer，只支持 IE ，通知浏览器，这段脚本代码将不会产生任何文档内容。

(2) async：html5 中新增的属性，它的作用是能够异步地下载和执行脚本，不因为加载脚本而阻塞页面的加载。一旦下载完毕就会立刻执行。

(3) 创建 script，插入到 DOM 中，加载完毕后 callBack

js 文件直接加载在文档的 head 标签里面，在写 js 文件的时候有时候获取一些文件对象的时候为空对象，这是由于文档结构还没有加载完，但是 js 文件已经加载完。也就是说虽然写了 js 语句来获取对象，但是由于 dom 结构还没有加载完成，因此获取到的是空对象，进一步测试发现在 firebug 的控制台下把赋值语句执行之后可以获得

对象，同理是因为在文档已经得到显示之后文档结构已经处于加载完成的状态，所以可以直接获取到对应的文档对象。
解决方法用两种：defer 和 async。
• defer="defer":
该属性用来通知浏览器，这段脚本代码将不会产生任何文档内容。例如 JavaScript 代码中的 document.write() 方法将不会起作用，浏览器遇到这样的代码将会忽略，并继续执行后面的代码。属性只能是 defer，与属性名相同。在 HTML 语法格式下，也允许不定义属性值，仅仅使用属性名。
• async="true/
false":该属性为 html5 中新增的属性，它的作用是能够异步地下载和执行脚本，不因为加载脚本而阻塞页面的加载。一旦下载完毕就会立刻执行。 script 中的 defer 属性默认情况下是 false 的，因此在使用时需要显式调用这一属性。 defer 既可用于载入 js 文件，也可用于行内脚本。 加上 defer 等于在页面完全在入后再执行，相当于 window.onload ，但应用上比 window.onload 更灵活！实际上 defer 更接近于 DomContentLoad。 事实上脚本执行于 onload 事件之前，即文档载入后即执行，不用等于包括图片在内的资源下载完毕。
• async 和 defer 一样，都不会阻塞其他资源下载，所以不会影响页面的加载，但在 async 的情况下，js 文档一旦下载完毕就会立刻执行，所以很有可能不是按照原本的顺序来执行，如果 js 有依赖性，就要注意了。
defer 和 async 的比较
• 相同点：
• 加载文件时不阻塞页面渲染；
• 对于 inline 的 script 无效；
• 使用这两个属性的脚本中不能调用 document.write 方法；
• 有脚本的 onload 的事件回调；
• 允许不定义属性值，仅仅使用属性名；
• 不同点：
• html 的版本 html4.0 中定义了 defer；html5.0 中定义了 async；这将造成由于浏览器版本的不同而对其支持的程度不同； • 执行时刻：每一个 async 属性的脚本都在它下载结束之后立刻执行，同时会在 window 的 load 事件之前执行。所以就有可能出现脚本执行顺序被打乱 的情况；每一个 defer 属性的脚本都是在页面解析完毕之后，按照原本的顺序执行，同时会在 document 的 DOMContentLoaded 之前执 行。 • 这两个属性会有三种可能的组合： • 如果 async 为 true，那么脚本在下载完成后异步执行。 • 如果 async 为 false，defer 为 true，那么脚本会在页面解析完毕之后执行。 • 如果 async 和 defer 都为 false，那么脚本会在页面解析中，停止页面解析，立刻下载并且执行。

十七、模块化怎么做？
一个模块就是实现特定功能的文件，有了模块，我们就可以更方便地使用别人的代码，想要什么功能，就加载什么模块。模块开发需要遵循一定的规范，否则就都乱套了。 根据 AMD 规范，我们可以使用 define 定义模块，使用 require 调用模块。
把模块写成一个对象，所有的模块成员都放到这个对象里面；使用"立即执行函数"（Immediately-Invoked Function Expression，IIFE），可以达到不暴露私有成员的目的

```
var module1 = (function(){
    var _count = 0;
    var m1 = function(){
        //...
    };
    var m2 = function(){
        //...
    };
    return {
        m1 : m1,
        m2 : m2
    };
})();
```

十八、Jquery 与 jQuery UI 有啥区别?

(1) jQuery 是一个 js 库，主要提供的功能是选择器，属性修改和事件绑定等等。

(2) jQuery UI 则是在 jQuery 的基础上，利用 jQuery 的扩展性，设计的插件。提供了一些常用的界面元素，诸如对话框、拖动行为、改变大小行为等等。

(3) jQuery 本身注重于后台，没有漂亮的界面，而 jQuery UI 则补充了前者的不足，他提供了华丽的展示界面，使人更容易接受。既有强大的后台，又有华丽的前台。jQuery UI 是 jQuery 插件，只不过专指由 jQuery 官方维护的 UI 方向的插件。

十九、Javascript 中 callee 和 caller 的作用?

caller 返回一个函数的引用，这个函数调用了当前的函数;callee 返回正在执行的函数本身的引用，它是 arguments 的一个属性

caller

caller 返回一个函数的引用，这个函数调用了当前的函数。

使用这个属性要注意:

1 这个属性只有当函数在执行时才有用

2 如果在 JavaScript 程序中，函数是由顶层调用的，则返回 null

```
functionName.caller: functionName是当前正在执行的函数。
    var a = function() {
    alert(a.caller);
    }
    var b = function() {
    a();
    }
    b();
上面的代码中，b调用了a，那么a.caller返回的是b的引用，结果如下:
    var b = function() {
    a();
    }
如果直接调用a(即a在任何函数中被调用，也就是顶层调用),返回null:
    var a = function() {
    alert(a.caller);
    }
    var b = function() {
    a();
    }
    //b();
    a();
输出结果: null
```

callee

callee 返回正在执行的函数本身的引用，它是 arguments 的一个属性

使用 callee 时要注意:

1 这个属性只有在函数执行时才有效

2 它有一个 length 属性，可以用来获得形参的个数，因此可以用来比较形参和实参个数是否一致，即比较 arguments.length 是否等于 arguments.callee.length

3 它可以用来递归匿名函数。

```
var a = function() {
alert(arguments.callee);
}
var b = function() {
a();
}
b();
a在b中被调用，但是它返回了a本身的引用，结果如下:
var a = function() {
alert(arguments.callee);
}
caller是返回一个对函数的引用，该函数调用了当前函数;
callee是返回正在被执行的function函数，也就是所指定的function对象的正文。
```

二十、flash 中 ActionScript2.0 和 ActionScript3.0 面向对象的异同

相同点：

1、都是 FLASH 脚本语言

2、都是用于开发 FLASH 交互应用的程序语言

不同点：

1、书写规范上，AS3 不再允许在按钮元件或者影片剪辑元件上添加动作代码。

2、点击或其它事件处理中，AS3 必须添加侦听才可以使用，去掉侦听才会停止，而 AS2 不是很严格。

3、很多类名的更换，如 AS2 中的_x,_y_parent 等都去掉了前面的下划线。

4、运行性能的改进，AS3 的运行性能要比 AS2 快十倍。

HTTP 部分

1、http 状态码有那些？分别代表是什么意思？

100-199 用于指定客户端应相应的某些动作。

200-299 用于表示请求成功。

300-399 用于已经移动的文件并且常被包含在定位头信息中指定新的地址信息。

400-499 用于指出客户端的错误。

400 1、语义有误，当前请求无法被服务器理解。401 当前请求需要用户验证 403 服务器已经理解请求，但是拒绝执行它。

500-599 用于支持服务器错误。 503 - 服务不可用

2、一次完整的 HTTP 事务是怎样的一个过程？

基本流程：

a. 域名解析

b. 发起 TCP 的 3 次握手

c. 建立 TCP 连接后发起 http 请求

d. 服务器端响应 http 请求，浏览器得到 html 代码

e. 浏览器解析 html 代码，并请求 html 代码中的资源

f. 浏览器对页面进行渲染呈现给用户

3、HTTPS 是如何实现加密？

https 是在 http 的基础上多了一次协议 ssl，该协议用来给传输的内容进行加密

https 加密流程：

1，客户端将 SSL 协议的版本号、加密算法的种类，产生的随机数 A 等信息传给服务器

2，服务器选择其中的一种组合作为加密方式，同时将自己的证书、公钥、另外一个随机数 B 一起传给客户端

3，客户端验证客户端返回的信息（包括证书，签名，域名等），验证成功，则生成对称加密密钥 S，用公钥加密后返回给服务器

4，服务器用私钥将这段密钥解密。得到对称密钥 S，并用此密钥加密一段握手消息返回给客户端

5，客户端收到握手消息，用对称密钥解密，验证成功，则握手成功。

单向加密和双向加密：

单向加密：服务端有一套（两份）证书：含公钥和私钥的 jks 文件（此文件自己保留），和只含公钥的 ser 文件，其中 ser 文件是要给客户端的

双向加密：除了具有单向加密的特性（服务端验证客户端），客户端也有一套（两份）证书来验证服务端消息，对于 android 来说，分为 bks 文件（含公钥和私钥，自己保留）、ser 文件（含公钥给服务端）

4、跨域请求资源的方法有哪些？

跨域：由于浏览器同源策略，凡是发送请求 url 的协议、域名、端口三者之间任意一与当前页面地址不同即为跨域。

同源策略： 一段脚本只能读取来自于同一来源的窗口和文档的属性，这里的同一来源指的是主机名、协议和端口号的组合

JSONP

这种方式主要是通过动态插入一个 script 标签。浏览器对 script 的资源引用没有同源限制，同时资源加载到页面后会立即执行（没有阻塞的情况下）。实际项目中 JSONP 通常用来获取 json 格式数据，这时前后端通常约定一个参数 callback，该参数的值，就是处理返回数据的函数名称。

缺点：

1、这种方式无法发送 post 请求（这里）

2、另外要确定 jsonp 的请求是否失败并不容易，大多数框架的实现都是结合超时时间来判定。

Proxy 代理

这种方式首先将请求发送给后台服务器，通过服务器来发送请求，然后将请求的结果传递给前端。 需要注意的是如果你代理的是 https 协议的请求，那么你的 proxy 首先需要信任该证书（尤其是自定义证书）或者忽略证书检查，否则你的请求无法成功。12306 就提供了一个鲜活的例子。

还需要注意一点，对于同一请求浏览器通常会从缓存中读取数据，我们有时候不想从缓存中读取，所以会加一个 preventCache 参数，这个时候请求 url 变成：url?preventCache=12345567....;这本身没有什么问题，问题出在当使用某些前端框架（比如 jquery）发送 proxy 代理请求时，请求 url 为 proxy?url，同时设置 preventCache：true，框架不能正确处理这个参数，结果发出去的请求变成 proxy?url&
preventCache=123456（正长应为 proxy?url?preventCache=12356）;后端截取后发送的请求为 url&
preventCache=123456，根本没有这个地址，所以你得不到正确结果。

CORS

这是现代浏览器支持跨域资源请求的一种方式。

当你使用 XMLHttpRequest 发送请求时，浏览器发现该请求不符合同源策略，会给该请求加一个请求头：Origin，后台进行一系列处理，如果确定接受请求则在返回结果中加入一个响应头：Access-Control-Allow-Origin;浏览器判断该相应头中是否包含 Origin 的值，如果有则浏览器会处理响应，我们就可以拿到响应数据，如果不包含浏览器直接驳回，这时我们无法拿到响应数据。

XDR

这是 IE8、IE9 提供的一种跨域解决方案，功能较弱只支持 get 跟 post 请求，而且对于协议不同的跨域是无能为力的，比如在 http 协议下发送 https 请求。

单向的数据请求，我们应该优先选择 JSONP 或者 window.name，双向通信我们采取 Cross Frame，在未与数据提供方没有达成通信协议的情况下我们也可以用 server proxy 的方式来抓取数据。

5、描述一下 cookies，sessionStorage 和 localStorage 的区别

sessionStorage 用于本地存储一个会话（session）中的数据，这些数据只有在同一个会话中的页面才能访问并且当会话结束后数据也随之销毁。因此 sessionStorage 不是一种持久化的本地存储，仅仅是会话级别的存储。

而 localStorage 用于持久化的本地存储，除非主动删除数据，否则数据是永远不会过期的。

web storage 和 cookie 的区别

Web Storage 的概念和 cookie 相似，区别是它是为了更大容量存储设计的。Cookie 的大小是受限的，并且每次你请求一个新的页面的时候 Cookie 都会被发送过去，这样无形中浪费了带宽，另外 cookie 还需要指定作用域，不可以跨域调用。

除此之外，Web Storage 拥有 setItem,getItem,removeItem,clear 等方法，不像 cookie 需要前端开发者自己封装 setCookie，getCookie。但是 Cookie 也是不可以或缺的：Cookie 的作用是与服务器进行交互，作为 HTTP 规范的一部分而存在 ，而 Web Storage 仅仅是为了在本地“存储”数据而生。

性能优化

1、前端开发的优化问题（看雅虎 14 条性能优化原则）

（1） 减少 http 请求次数：CSS Sprites, JS、CSS 源码压缩、图片大小控制合适；网页 Gzip，CDN 托管，data 缓存 ，图片服务器。

（2） 前端模板 JS+数据，减少由于 HTML 标签导致的带宽浪费，前端用变量保存 AJAX 请求结果，每次操作本地变量，不用请求，减少请求次数。

（3） 用 innerHTML 代替 DOM 操作，减少 DOM 操作次数，优化 javascript 性能。

（4） 当需要设置的样式很多时设置 className 而不是直接操作 style。

（5） 少用全局变量、缓存 DOM 节点查找的结果。减少 IO 读取操作。

（6） 避免使用 CSS Expression（css 表达式)又称 Dynamic properties(动态属性)。

（7） 图片预加载，将样式表放在顶部，将脚本放在底部 加上时间戳。

（8） 避免在页面的主体布局中使用 table，table 要等其中的内容完全下载之后才会显示出来，显示比 div+css 布局慢。

2、如何提高网页的运行速度

*基于 Class 的选择性的性能相对于 Id 选择器开销很大，因为需遍历所有 DOM 元素。

*频繁操作的 DOM，先缓存起来再操作。用 Jquery 的链式调用更好。

比如：var str=$("a").attr("href");

*for (var i = size; i < arr.length; i++) {}

for 循环每一次循环都查找了数组 (arr) 的.length 属性，在开始循环的时候设置一个变量来存储这个数字，可以让循环跑得更快：
for (var i = size, length = arr.length; i < length; i++) {}

其它相关知识点
1、什么叫优雅降级和渐进增强？
优雅降级：
Web 站点在所有新式浏览器中都能正常工作，如果用户使用的是老式浏览器，则代码会检查以确认它们是否能正常工作。由于 IE 独特的盒模型布局问题，针对不同版本的 IE 的 hack 实践过优雅降级了,为那些无法支持功能的浏览器增加候选方案，使之在旧式浏览器上以某种形式降级体验却不至于完全失效.
渐进增强：
从被所有浏览器支持的基本功能开始，逐步地添加那些只有新式浏览器才支持的功能,向页面增加无害于基础浏览器的额外样式和功能的。当浏览器支持时，它们会自动地呈现出来并发挥作用。
2、简述同步和异步的区别
同步是阻塞模式，异步是非阻塞模式。
同步就是指一个进程在执行某个请求的时候，若该请求需要一段时间才能返回信息，那么这个进程将会一直等待下去，直到收到返回信息才继续执行下去；
异步是指进程不需要一直等下去，而是继续执行下面的操作，不管其他进程的状态。当有消息返回时系统会通知进程进行处理，这样可以提高执行的效率。
3、常用的库有哪些？常用的前端开发工具？
* 使用率较高的框架有 jQuery、YUI、Prototype、Dojo、Ext.js、Mootools 等。尤其是 jQuery，超过 91%。
*轻量级框架有 Modernizr、underscore.js、backbone.js、Raphael.js 等。 （理解这些框架的功能、性能、设计原理）
* 常用开发工具：Sublime Text 、Eclipse、Notepad、Firebug、HttpWatch、Yslow。