**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.xhtml**和**html**有什么区别

**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。

**XHTML** 元素必须被关闭。

　　标签名必须用小写字母。

**XHTML** 文档必须拥有根元素。

**3.Doctype?** 严格模式与混杂模式**-**如何触发这两种模式，区分它们有何意义**?**

　　用于声明文档使用那种规范**(html/Xhtml)**一般为严格过度基于框架的**html**文档

　　加入**XMl**声明可触发，解析方式更改为**IE5.5** 拥有**IE5.5**的**bug**

**4.**行内元素有哪些**?**块级元素有哪些**?CSS**的盒模型**?**

　　块级元素：**div p h1 h2 h3 h4 form ul**

　　行内元素**: a b br i span input select**

**Css**盒模型**:**内容，**border ,margin**，**padding**

**5.CSS**引入的方式有哪些**? link**和**@import**的区别是**?**

　　内联内嵌外链导入

　　区别：同时加载

　　前者无兼容性，后者**CSS2.1**以下浏览器不支持

**Link** 支持使用**javascript**改变样式，后者不可

**6.CSS**选择符有哪些**?**哪些属性可以继承**?**优先级算法如何计算**?**内联和**important**哪个优先级高**?**

　　标签选择符类选择符 **id**选择符

　　继承不如指定 **Id>class>**标签选择

　　后者优先级高

**7.**前端页面有哪三层构成，分别是什么**?**作用是什么**?**

　　结构层 **Html** 表示层 **CSS** 行为层 **js**

**8.css**的基本语句构成是**?**

　　选择器**{**属性**1:**值**1;**属性**2:**值**2;……}**

**9.**你做的页面在哪些流览器测试过**?**这些浏览器的内核分别是什么**?**

**Ie(Ie**内核**)** 火狐**(Gecko)** 谷歌**(webkit) opear(Presto)**

**10.**写出几种**IE6 BUG**的解决方法

**1.**双边距**BUG float**引起的使用**display**

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

**3.**超链接**hover** 点击后失效使用正确的书写顺序 **link visited hover active**

**4.Ie z-index**问题给父级添加**position:relative**

**5.Png** 透明使用**js**代码改

**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.ie 6** 不支持**!important**

**11.img**标签上**title**与**alt**属性的区别是什么**?**

**Alt** 当图片不显示是用文字代表。

**Title** 为该属性提供信息

**12.**描述**css reset**的作用和用途。

**Reset**重置浏览器的**css**默认属性浏览器的品种不同，样式不同，然后重置，让他们统一

**13.**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

**14.**浏览器标准模式和怪异模式之间的区别是什么**?**

　　盒子模型渲染模式的不同

　　使用 **window.top.document.compatMode** 可显示为什么模式

**15.**你如何对网站的文件和资源进行优化**?**期待的解决方案包括：

　　文件合并

　　文件最小化**/**文件压缩

　　使用**CDN**托管

　　缓存的使用

**16.**什么是语义化的**HTML?**

　　直观的认识标签对于搜索引擎的抓取有好处

**17.**清除浮动的几种方式，各自的优缺点

**1.**使用空标签清除浮动 **clear:both(**理论上能清楚任何标签，，，增加无意义的标签**)**

**2.**使用**overflow:auto(**空标签元素清除浮动而不得不增加无意代码的弊端**,,**使用**zoom:1**用于兼容**IE)**

**3.**是用**afert**伪元素清除浮动**(**用于非**IE**浏览器**)**

**18.css hack**

**<!--[if IE 6]--><![end if]-->**

**\_marging \\IE 6**

**+margin \\IE 7**

**Marging:0 auto \9** 所有**Ie**

**Margin \0 \\IE 8**

**JavaScript**：

**1.** 介绍一下**isNaN**函数？

如果参数不是数字类型的话**isNaN**返回**true** （函数名称：**isnan** 函数功能：判断数组的元素是否是**NaN**。）

**2. javascript**都支持哪些布尔型的操作？

**&&, ||**和！（**&&** 逻辑与，**||** 逻辑或，！逻辑非）

**3. “1″+2+4**在**javascript**中得到什么？

因为**“1”**是**String**类型，所以所有的操作数都是**String,** 结果为**124**

**4. 2+5+”8″**得到的结果是什么？

因为**2**和**5**是数字类型，**8**是字符串，所以得到的结果是**78**

**5. javascript**有几种**loop**（循环）方法？

**for, while, do-while** ，**for...in** 语句用于遍历数组或者对象的属性（对数组或者对象的属性进行循环操作）。

**6. javascript**如何创建一个新的对象？

**var obj = new Object()** 或者 **var obj = {}**

**7.** 如何为一个对象属性赋值？

**obj["age"]=17** 或者**obj.age=17**

**8.** 如何为一个数组添加一个值？

**arr[arr.length]= value;**

**9.javascript**基本数据类型 **/ javascript**中的**typeof**返回哪些数据类型

**Number** 类型、**Boolean** 类型、**Object** 类型、**String**类型、**Null**、 **Undefined** 类型 **/ Object number function boolean undefined string**

**10.**例举**3**种强制类型转换和**2**种隐式类型转换**?**

强制（**parseInt,parseFloat,number**）隐式（**== – ===**）

**11.**数组方法**pop() push() unshift() shift()**

**Push()**尾部添加 **pop()**尾部删除 **Unshift()**头部添加 **shift()**头部删除

**12.**闭包是什么，有什么特性，对页面有什么影响

闭包就是能够读取其他函数内部变量的函数。

**13.**添加删除替换插入到某个接点的方法

**obj.appendChild() obj.innersetBefore() obj.replaceChild() obj.removeChild()**

**14.javascript**的本地对象，内置对象和宿主对象

本地对象为**array obj regexp**等可以**new**实例化

内置对象为**gload Math** 等不可以实例化的

宿主为浏览器自带的**document,window** 等

**15.**介绍一下**eval()** 函数

**eval()** 函数可计算某个字符串，并执行其中的的 **JavaScript** 代码。

**16.**介绍一下**JSON**

**JSON**：**JavaScript** 对象表示法（**JavaScript Object Notation**）。

**JSON** 是存储和交换文本信息的语法。类似 **XML**。

**JSON** 比 **XML** 更小、更快，更易解析。

**17.jQuery UI** 与 **jquery** 的主要区别是：

**(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。

**html+css:**

**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.xhtml**和**html**有什么区别**:**

**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。

**XHTML** 元素必须被关闭。

标签名必须用小写字母。

**XHTML** 文档必须拥有根元素。

**3.**行内元素有哪些**?**块级元素有哪些**?CSS**的盒模型**?**

块级元素：**div p h1 h2 h3 h4 form ul**

行内元素**: a b br i span input select**

**Css**盒模型**:**内容，**border ,margin**，**padding**

**4.**前端页面有哪三层构成，分别是什么**?**作用是什么**?**

结构层 **Html** 表示层 **CSS** 行为层 **js**

**5.**写出几种**IE6 BUG**的解决方法**:**

**[list][\*]**双边距**BUG float**引起的使用**display**

**[\*]3**像素问题使用**float**引起的使用**dislpay:inline -3px**

**[\*]**超链接**hover** 点击后失效使用正确的书写顺序 **link visited hover active**

**[\*]Ie z-index**问题给父级添加**position:relative**

**[\*]Png** 透明使用**js**代码改

**[\*]Min-height** 最小高度！**Important** 解决**’**

**[\*]select** 在**ie6**下遮盖使用**iframe**嵌套

**[\*]**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**）

**[/list]6.**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

**7.**你如何对网站的文件和资源进行优化**?**期待的解决方案包括：

**[list][\*]** 文件合并

**[\*]** 文件最小化**/**文件压缩

**[\*]** 使用**CDN**托管

**[\*]** 缓存的使用

**[/list]8.**清除浮动的几种方式，各自的优缺点：

**[list][\*]**使用空标签清除浮动 **clear:both**（理论上能清楚任何标签，，，增加无意义的标签）

**[\*]**使用**overflow:auto**（空标签元素清除浮动而不得不增加无意代码的弊端**,,**使用**zoom:1**用于兼容**IE**）

**[\*]**是用**afert**伪元素清除浮动**(**用于非**IE**浏览器**)**

**[/list]9.**浏览器内核

**1**、**Trident**内核：**IE**最先开发或使用的，也称**IE**内核，**360**浏览器使用的也是**IE**内核；

**2**、**Webkit**内核：谷歌**chrome**浏览器最先开发或使用，也叫谷歌内核，枫树浏览器、太阳花使用的也是谷歌内核；

**3**、**Gecko**内核： **Netscape6**开始采用的内核，后来的**Mozilla FireFox (**火狐浏览器**)** 也采用了该内核，**K-Meleon**浏览器也是使用这种内核；

**4**、**Presto**内核：目前只有**Opera**浏览器采用该内核

**1.**写一下文档声明的格式

**html 5**的文档声明格式是：

**<!DOCTYPE html>**

**html 4** 的文档声明格式

**<!DOCTYPE HTML PUbLIC "-//W3C//DTD HTML 4.01 // EN" "http://www.w3/org/TR/html4/strict.tdt">**

**<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">**

**2.**块级元素与行内元素的区别；

在标准的文档流里面，块级元素的特点是：

**1.**独占一行；

**2.**高度，行高，以及外边距和内边距都可以控制

**3.**宽度如果没有设定的情况下，始终和浏览器的宽度一样，与内容无关

**4.**可以容纳内敛元素，和其他块级元素

行内元素的特点：

**1.**和其他元素都在一行

**2.**高，行高及外边距，和内边距部分可以改变

**3.**宽度至于内容有关

**4.**行内元素只能够容纳文本，或者其他行内元素。如果不转换成块级元素的话，

不可以设置宽高，其宽高随着内容的增加，高度随着字体的大小而改变

内联元素可以设置外边界，但是外边界不对上下起作用，只能够对左右起作用，也可以设置内边界，

但是内边界在**ie6**中不对上下起作用，只能对左右起作用

**3.**列举常见的**css hack** 技巧

**css hack** 是为了解决不同浏览器对 **css** 解析结果的不同，而采取的一种措施

**css hack** 常见的有三种形式； **css** 属性**hack**、 **css** 选择符**hack** 以及**IE**条件注释的**hack** ，**hack**主要针对的就是**IE** 浏览器

熟悉级**hack**：

**ie6** 能够识别下划线**"\_"** 和星号**"\*"**

**ie7** 能够识别**"\*",**不能够识别**"\_"**

**IE 6** 不能够识别！**important**

选择符**hack**：

**1.ie 6**能够识别 **\*html.class{}**

**ie 7**能够识别 **\*+html.class{}**

注释级**hack**

**<!--[if IE]>**此处内容只有**IE**可见**<![endif]-->**

**<!--[if IE 6]>**此处内容只有**IE6.0**可见**<![endif]-->**

**<!--[if IE 7]>**此处内容只有**IE7.0**可见**<![endif]-->**

**<!--[if !IE 7]>**此处内容只有**IE7**不能识别，其他版本都能识别，当然要在**IE5**以上。**<![endif]-->**

**<!--[if gt IE 6]> IE6**以上版本可识别**,IE6**无法识别 **<![endif]-->**

**4.**列举一些**css** 浏览器兼容性问题

**1**、**ie 6** 浮动时产生的双倍间距的问题解决方案， **display**：**inline**

**display** 的两个熟悉 **block** ，**inline** 两个元素

**block**的特点是总是在新行上开始**,**高度**,**宽度**,**行高**,**边距都可以控制**(**块元素**)**

**inline** 的特点是，和其他元素在同一行，不可控制（内嵌元素）

**#box{ display:block;} //**可以为内嵌元素模拟为块元素 **display:inline; //**实现同一行排列的效果 **diplay:table; /**

**2**、**IE**浏览器的宽度和高度的问题， **IE**不能识别 **min-**这个定义，但实际上他把正常的**width** 和**height** 当做**min** 的情况

来使用这样的话，就会出现问题，如果只用宽度和高度，正常的浏览器这两个值就不会变，

如果只用**min-width**和**min-height**的话，**IE**下面根本等于没有设置宽度和高度。

比如要设置背景图片，这个宽度是比较重要的。要解决这个问题，可以这样： **#box{ width: 80px; height: 35px;}**

**html>body #box{ width: auto; height: auto; min-width: 80px; min-height: 35px;}**

**3**、**div**浮动会让**IE**文本产生 **3px** 的**bug** 左边对象浮动，右边采用外补丁的左边距来定位，右边对象内的

微博会离左边有 **3px** 的间距，

**#box{ float:left; width:800px;}**

**#left{ float:left; width:50%;}**

**#right{ width:50%;}**

**\*html #left{ margin-right:-3px; //**这句是关键**}**

**<div id="box"> <div id="left"></div> <div id="right"></div> </div>**

**5.** 列举**css 3**都有哪些新内容

**css3** 的圆角 **border-radius**

文字阴影效果**; font-effect**

多背景图

**@font-face** ，　旋转**transform**　　　渐变

６**.**　提升网页加载速度和网页性能的方法

　１**.**减少**http** 请求，网站建设中过多的图片， **css script flash** 会增加内容加载的请求

负担，因此建议减少此类元素的使用，使用**css** 合并图片，避免直接插入原图

**2.** 使用 **gzip** 压缩文档增加网页加载速度，同时也可以降低网站的带宽流量的消耗

**3.** 设置合理的过期时间，尽可能减少**cookie** 体积，能够很好的提高页面的加载效率，对静态页

组件的**cookie** 读取是一种资源的浪费

**4.** 减少图片的体积

**5.** 样式表文件放在外部调用，把页面中需要用到的**javascript** 和 **css** 样式表文件放在另外的文件夹中，

这样一来在打开网站其它页面时就会缓存这些调用的文件，从而加快网站页面的加载速度。

**6**、减少域名**dns** 查找时间

**7.**如何实现网页在不同设备上进行等比缩放

采取响应式页面布局方式

在网页代码的头部加上一行 **viewport** 元标签

**<meta name="viewport" content="width=device-width,initial-scale=1"/>**

**<!–[if lt IE 9]><script src=”http://css3-mediaqueries-js.googlecode.com/svn/trunk/css3-mediaqueries.js”></script><![endif]–>**

**8:**

**alert(typeof(null));//object**

**alert(typeof(undefined));//undefined**

**alert(typeof(NaN));//number**

**alert(NaN==NaN);//false**

**var str="123abc"; alert(typeof(str++));//number**

**alert(str);//NaN**

**10.**解析**jsonp**的原理，以及**js** 的同源策略

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**，这项技术可以解决跨域提交**Ajax**请求的问题

**JSONP**跨域**GET**请求是一个常用的解决方案

**JavaScript** 的同源策略。这一策略对于**JavaScript**代码能够访问的页面内容做了很重要的限制，

即**JavaScript**只能访问与包含它的文档在同一域下的内容 **ajax** 不能够跨域请求，**xmlhtmmprequerst**

**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。

**js** 实现一个类的继承

**var Chinese = function(name, nation){**

**//**继承，需要调用父类的构造函数，可以用**call**调用，**this**指向**Chinese**

**//**使**Person**在此作用域上，才可以调用**Person**的成员

**Person.call(this,name);**

**this.nation = nation;**

**};**

**Chinese.prototype = Person.prototype;**

**//**这里不可和以前一样，因为覆盖掉了**prototype**属性

**//Chinese.prototype = {**

**// getNation : function(){**

**// return this.nation;**

**// }**

**//};**

**//**以后的方法都需要这样加

**Chinese.prototype.getNation = function(){**

**return this.nation;**

**};**

**var c = new Chinese("liyatang","China");**

**alert(c.getName());// liyatang**

**f=function(){return false};**

**g=function(){return true};**

**(function(){**

**if(g()&&[]==![]){**

**f=function(){return false};**

**function g(){return false};**

**}**

**})();**

**alert(f()); //false**

常见排序算法的时间复杂度**,**空间复杂度

前端需要注意哪些**SEO**

**-** 合理的**title**、**description**、**keywords**：搜索对着三项的权重逐个减小，**title**值强调重点即可，重要关键词出现不要超过**2**次，而且要靠前，不同页面**title**要有所不同；**description**把页面内容高度概括，长度合适，不可过分堆砌关键词，不同页面**description**有所不同；**keywords**列举出重要关键词即可

**-** 语义化的**HTML**代码，符合**W3C**规范：语义化代码让搜索引擎容易理解网页

**-** 重要内容**HTML**代码放在最前：搜索引擎抓取**HTML**顺序是从上到下，有的搜索引擎对抓取长度有限制，保证重要内容一定会被抓取

**-** 重要内容不要用**js**输出：爬虫不会执行**js**获取内容

**-** 少用**iframe**：搜索引擎不会抓取**iframe**中的内容

**-** 非装饰性图片必须加**alt**

**-** 提高网站速度：网站速度是搜索引擎排序的一个重要指标

**web**开发中会话跟踪的方法有哪些

**- cookie**

**- session**

**- url**重写

**-** 隐藏**input**

**- ip**地址

**<img>**的**title**和**alt**有什么区别

**- title**是**global attributes**之一，用于为元素提供附加的**advisory information**。通常当鼠标滑动到元素上的时候显示。

**- alt**是**<img>**的特有属性，是图片内容的等价描述，用于图片无法加载时显示、读屏器阅读图片。可提图片高可访问性，除了纯装饰图片外都必须设置有意义的值，搜索引擎会重点分析。

**doctype**是什么**,**举例常见**doctype**及特点

**- <!doctype>**声明必须处于**HTML**文档的头部，在**<html>**标签之前，**HTML5**中不区分大小写

**- <!doctype>**声明不是一个**HTML**标签，是一个用于告诉浏览器当前**HTMl**版本的指令

**-** 现代浏览器的**html**布局引擎通过检查**doctype**决定使用兼容模式还是标准模式对文档进行渲染，一些浏览器有一个接近标准模型。

**-** 在**HTML4.01**中**<!doctype>**声明指向一个**DTD**，由于**HTML4.01**基于**SGML**，所以**DTD**指定了标记规则以保证浏览器正确渲染内容

**- HTML5**不基于**SGML**，所以不用指定**DTD**

常见**dotype**：

**- HTML4.01 strict**：不允许使用表现性、废弃元素（如**font**）以及**frameset**。声明：**<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">**

**- HTML4.01 Transitional:**允许使用表现性、废弃元素（如**font**），不允许使用**frameset**。声明：**<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">**

**- HTML4.01 Frameset:**允许表现性元素，废气元素以及**frameset**。声明：**<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">**

**- XHTML1.0 Strict:**不使用允许表现性、废弃元素以及**frameset**。文档必须是结构良好的**XML**文档。声明：**<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">**

**- XHTML1.0 Transitional:**允许使用表现性、废弃元素，不允许**frameset**，文档必须是结构良好的**XMl**文档。声明：**<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">**

**- XHTML 1.0 Frameset:**允许使用表现性、废弃元素以及**frameset**，文档必须是结构良好的**XML**文档。声明：**<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">**

**- HTML 5: <!doctype html>**

**HTML**全局属性**(global attribute)**有哪些

参考资料：**MDN: html global attribute**或者**W3C HTML global-attributes**

**- accesskey:**设置快捷键，提供快速访问元素如**aaa**在**windows**下的**firefox**中按**alt + shift + a**可激活元素

**- class:**为元素设置类标识，多个类名用空格分开，**CSS**和**javascript**可通过**class**属性获取元素

**- contenteditable:** 指定元素内容是否可编辑

**- contextmenu:** 自定义鼠标右键弹出菜单内容

**- data-\*:** 为元素增加自定义属性

**- dir:** 设置元素文本方向

**- draggable:** 设置元素是否可拖拽

**- dropzone:** 设置元素拖放类型： **copy, move, link**

**- hidden:** 表示一个元素是否与文档。样式上会导致元素不显示，但是不能用这个属性实现样式效果

**- id:** 元素**id**，文档内唯一

**- lang:** 元素内容的的语言

**- spellcheck:** 是否启动拼写和语法检查

**- style:** 行内**css**样式

**- tabindex:** 设置元素可以获得焦点，通过**tab**可以导航

**- title:** 元素相关的建议信息

**- translate:** 元素和子孙节点内容是否需要本地化

什么是**web**语义化**,**有什么好处

**web**语义化是指通过**HTML**标记表示页面包含的信息，包含了**HTML**标签的语义化和**css**命名的语义化。 **HTML**标签的语义化是指：通过使用包含语义的标签（如**h1-h6**）恰当地表示文档结构 **css**命名的语义化是指：为**html**标签添加有意义的**class**，**id**补充未表达的语义，如**Microformat**通过添加符合规则的**class**描述信息为什么需要语义化：

**-** 去掉样式后页面呈现清晰的结构

**-** 盲人使用读屏器更好地阅读

**-** 搜索引擎更好地理解页面，有利于收录

**-** 便团队项目的可持续运作及维护

**HTTP method**

**-** 一台服务器要与**HTTP1.1**兼容，只要为资源实现**GET**和**HEAD**方法即可

**- GET**是最常用的方法，通常用于请求服务器发送某个资源。

**- HEAD**与**GET**类似，但服务器在响应中值返回首部，不返回实体的主体部分

**- PUT**让服务器用请求的主体部分来创建一个由所请求的**URL**命名的新文档，或者，如果那个**URL**已经存在的话，就用干这个主体替代它

**- POST**起初是用来向服务器输入数据的。实际上，通常会用它来支持**HTML**的表单。表单中填好的数据通常会被送给服务器，然后由服务器将其发送到要去的地方。

**- TRACE**会在目的服务器端发起一个环回诊断，最后一站的服务器会弹回一个**TRACE**响应并在响应主体中携带它收到的原始请求报文。**TRACE**方法主要用于诊断，用于验证请求是否如愿穿过了请求**/**响应链。

**- OPTIONS**方法请求**web**服务器告知其支持的各种功能。可以查询服务器支持哪些方法或者对某些特殊资源支持哪些方法。

**- DELETE**请求服务器删除请求**URL**指定的资源

从浏览器地址栏输入**url**到显示页面的步骤**(**以**HTTP**为例**)**

**-** 在浏览器地址栏输入**URL**

**-** 浏览器查看缓存，如果请求资源在缓存中并且新鲜，跳转到转码步骤

**-** 如果资源未缓存，发起新请求

**-** 如果已缓存，检验是否足够新鲜，足够新鲜直接提供给客户端，否则与服务器进行验证。

**-** 检验新鲜通常有两个**HTTP**头进行控制**Expires**和**Cache-Control**：

**- HTTP1.0**提供**Expires**，值为一个绝对时间表示缓存新鲜日期

**- HTTP1.1**增加了**Cache-Control: max-age=,**值为以秒为单位的最大新鲜时间

**-** 浏览器解析**URL**获取协议，主机，端口，**path**

**-** 浏览器组装一个**HTTP**（**GET**）请求报文

**-** 浏览器获取主机**ip**地址，过程如下：

**-** 浏览器缓存

**-** 本机缓存

**- hosts**文件

**-** 路由器缓存

**- ISP DNS**缓存

**- DNS**递归查询（可能存在负载均衡导致每次**IP**不一样）

**-** 打开一个**socket**与目标**IP**地址，端口建立**TCP**链接，三次握手如下：

**-** 客户端发送一个**TCP**的**SYN=1**，**Seq=X**的包到服务器端口

**-** 服务器发回**SYN=1**， **ACK=X+1**， **Seq=Y**的响应包

**-** 客户端发送**ACK=Y+1**， **Seq=Z**

**- TCP**链接建立后发送**HTTP**请求

**-** 服务器接受请求并解析，将请求转发到服务程序，如虚拟主机使用**HTTP Host**头部判断请求的服务程序

**-** 服务器检查**HTTP**请求头是否包含缓存验证信息如果验证缓存新鲜，返回**304**等对应状态码

**-** 处理程序读取完整请求并准备**HTTP**响应，可能需要查询数据库等操作

**-** 服务器将响应报文通过**TCP**连接发送回浏览器

**-** 浏览器接收**HTTP**响应，然后根据情况选择关闭**TCP**连接或者保留重用，关闭**TCP**连接的四次握手如下：

**-** 主动方发送**Fin=1**， **Ack=Z**， **Seq= X**报文

**-** 被动方发送**ACK=X+1**， **Seq=Z**报文

**-** 被动方发送**Fin=1**， **ACK=X**， **Seq=Y**报文

**-** 主动方发送**ACK=Y**， **Seq=X**报文

**-** 浏览器检查响应状态吗：是否为**1XX**，**3XX**， **4XX**， **5XX**，这些情况处理与**2XX**不同

**-** 如果资源可缓存，进行缓存

**-** 对响应进行解码（例如**gzip**压缩）

**-** 根据资源类型决定如何处理（假设资源为**HTML**文档）

**-** 解析**HTML**文档，构件**DOM**树，下载资源，构造**CSSOM**树，执行**js**脚本，这些操作没有严格的先后顺序，以下分别解释

**-** 构建**DOM**树：

**- Tokenizing**：根据**HTML**规范将字符流解析为标记

**- Lexing**：词法分析将标记转换为对象并定义属性和规则

**- DOM construction**：根据**HTML**标记关系将对象组成**DOM**树

**-** 解析过程中遇到图片、样式表、**js**文件，启动下载

**-** 构建**CSSOM**树：

**- Tokenizing**：字符流转换为标记流

**- Node**：根据标记创建节点

**- CSSOM**：节点创建**CSSOM**树

**-** 根据**DOM**树和**CSSOM**树构建渲染树**:**

**-** 从**DOM**树的根节点遍历所有可见节点，不可见节点包括：**1**）**script,meta**这样本身不可见的标签。**2)**被**css**隐藏的节点，如**display: none**

**-** 对每一个可见节点，找到恰当的**CSSOM**规则并应用

**-** 发布可视节点的内容和计算样式

**- js**解析如下：

**-** 浏览器创建**Document**对象并解析**HTML**，将解析到的元素和文本节点添加到文档中，此时**document.readystate**为**loading**

**- HTML**解析器遇到没有**async**和**defer**的**script**时，将他们添加到文档中，然后执行行内或外部脚本。这些脚本会同步执行，并且在脚本下载和执行时解析器会暂停。这样就可以用**document.write()**把文本插入到输入流中。同步脚本经常简单定义函数和注册事件处理程序，他们可以遍历和操作**script**和他们之前的文档内容

**-** 当解析器遇到设置了**async**属性的**script**时，开始下载脚本并继续解析文档。脚本会在它下载完成后尽快执行，但是解析器不会停下来等它下载。异步脚本禁止使用**document.write()**，它们可以访问自己**script**和之前的文档元素

**-** 当文档完成解析，**document.readState**变成**interactive**

**-** 所有**defer**脚本会按照在文档出现的顺序执行，延迟脚本能访问完整文档树，禁止使用**document.write()**

**-** 浏览器在**Document**对象上触发**DOMContentLoaded**事件

**-** 此时文档完全解析完成，浏览器可能还在等待如图片等内容加载，等这些内容完成载入并且所有异步脚本完成载入和执行，**document.readState**变为**complete,window**触发**load**事件

**-** 显示页面（**HTML**解析过程中会逐步显示页面）

**HTTP request**报文结构是怎样的

**rfc2616**中进行了定义：

**-** 首行是**Request-Line**包括：请求方法，请求**URI**，协议版本，**CRLF**

**-** 首行之后是若干行请求头，包括**general-header**，**request-header**或者**entity-header**，每个一行以**CRLF**结束

**-** 请求头和消息实体之间有一个**CRLF**分隔

**-** 根据实际请求需要可能包含一个消息实体一个请求报文例子如下：

**GET /Protocols/rfc2616/rfc2616-sec5.html HTTP/1.1**

**Host: www.w3.org**

**Connection: keep-alive**

**Cache-Control: max-age=0**

**Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,\*/\*;q=0.8**

**User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/35.0.1916.153 Safari/537.36**

**Referer: https://www.google.com.hk/**

**Accept-Encoding: gzip,deflate,sdch**

**Accept-Language: zh-CN,zh;q=0.8,en;q=0.6**

**Cookie: authorstyle=yes**

**If-None-Match: "2cc8-3e3073913b100"**

**If-Modified-Since: Wed, 01 Sep 2004 13:24:52 GMT**

**name=qiu&age=25**

**HTTP response**报文结构是怎样的

**rfc2616**中进行了定义：

**-** 首行是状态行包括：**HTTP**版本，状态码，状态描述，后面跟一个**CRLF**

**-** 首行之后是若干行响应头，包括：通用头部，响应头部，实体头部

**-** 响应头部和响应实体之间用一个**CRLF**空行分隔

**-** 最后是一个可能的消息实体响应报文例子如下：

**HTTP/1.1 200 OK**

**Date: Tue, 08 Jul 2014 05:28:43 GMT**

**Server: Apache/2**

**Last-Modified: Wed, 01 Sep 2004 13:24:52 GMT**

**ETag: "40d7-3e3073913b100"**

**Accept-Ranges: bytes**

**Content-Length: 16599**

**Cache-Control: max-age=21600**

**Expires: Tue, 08 Jul 2014 11:28:43 GMT**

**P3P: policyref="http://www.w3.org/2001/05/P3P/p3p.xml"**

**Content-Type: text/html; charset=iso-8859-1**

**{"name": "qiu", "age": 25}**

如何进行网站性能优化

雅虎**Best Practices for Speeding Up Your Web Site**：

**- content**方面

**-** 减少**HTTP**请求：合并文件、**CSS**精灵、**inline Image**

**-** 减少**DNS**查询：**DNS**查询完成之前浏览器不能从这个主机下载任何任何文件。方法：**DNS**缓存、将资源分布到恰当数量的主机名，平衡并行下载和**DNS**查询

**-** 避免重定向：多余的中间访问

**-** 使**Ajax**可缓存

**-** 非必须组件延迟加载

**-** 未来所需组件预加载

**-** 减少**DOM**元素数量

**-** 将资源放到不同的域下：浏览器同时从一个域下载资源的数目有限，增加域可以提高并行下载量

**-** 减少**iframe**数量

**-** 不要**404**

**- Server**方面

**-** 使用**CDN**

**-** 添加**Expires**或者**Cache-Control**响应头

**-** 对组件使用**Gzip**压缩

**-** 配置**ETag**

**- Flush Buffer Early**

**- Ajax**使用**GET**进行请求

**-** 避免空**src**的**img**标签

**- Cookie**方面

**-** 减小**cookie**大小

**-** 引入资源的域名不要包含**cookie**

**- css**方面

**-** 将样式表放到页面顶部

**-** 不使用**CSS**表达式

**-** 使用不使用**@import**

**-** 不使用**IE**的**Filter**

**- Javascript**方面

**-** 将脚本放到页面底部

**-** 将**javascript**和**css**从外部引入

**-** 压缩**javascript**和**css**

**-** 删除不需要的脚本

**-** 减少**DOM**访问

**-** 合理设计事件监听器

**-** 图片方面

**-** 优化图片：根据实际颜色需要选择色深、压缩

**-** 优化**css**精灵

**-** 不要在**HTML**中拉伸图片

**-** 保证**favicon.ico**小并且可缓存

**-** 移动方面

**-** 保证组件小于**25k**

**- Pack Components into a Multipart Document**

什么是渐进增强

渐进增强是指在**web**设计时强调可访问性、语义化**HTML**标签、外部样式表和脚本。保证所有人都能访问页面的基本内容和功能同时为高级浏览器和高带宽用户提供更好的用户体验。核心原则如下**:**

**-** 所有浏览器都必须能访问基本内容

**-** 所有浏览器都必须能使用基本功能

**-** 所有内容都包含在语义化标签中

**-** 通过外部**CSS**提供增强的布局

**-** 通过非侵入式、外部**javascript**提供增强功能

**- end-user web browser preferences are respected**

**HTTP**状态码及其含义

参考**RFC 2616**

**- 1XX**：信息状态码

**- 100 Continue**：客户端应当继续发送请求。这个临时相应是用来通知客户端它的部分请求已经被服务器接收，且仍未被拒绝。客户端应当继续发送请求的剩余部分，或者如果请求已经完成，忽略这个响应。服务器必须在请求万仇向客户端发送一个最终响应

**- 101 Switching Protocols**：服务器已经理解力客户端的请求，并将通过**Upgrade**消息头通知客户端采用不同的协议来完成这个请求。在发送完这个响应最后的空行后，服务器将会切换到**Upgrade**消息头中定义的那些协议。

**- 2XX**：成功状态码

**- 200 OK**：请求成功，请求所希望的响应头或数据体将随此响应返回

**- 201 Created**：

**- 202 Accepted**：

**- 203 Non-Authoritative Information**：

**- 204 No Content**：

**- 205 Reset Content**：

**- 206 Partial Content**：

**- 3XX**：重定向

**- 300 Multiple Choices**：

**- 301 Moved Permanently**：

**- 302 Found**：

**- 303 See Other**：

**- 304 Not Modified**：

**- 305 Use Proxy**：

**- 306** （**unused**）：

**- 307 Temporary Redirect**：

**- 4XX**：客户端错误

**- 400 Bad Request:**

**- 401 Unauthorized:**

**- 402 Payment Required:**

**- 403 Forbidden:**

**- 404 Not Found:**

**- 405 Method Not Allowed:**

**- 406 Not Acceptable:**

**- 407 Proxy Authentication Required:**

**- 408 Request Timeout:**

**- 409 Conflict:**

**- 410 Gone:**

**- 411 Length Required:**

**- 412 Precondition Failed:**

**- 413 Request Entity Too Large:**

**- 414 Request-URI Too Long:**

**- 415 Unsupported Media Type:**

**- 416 Requested Range Not Satisfiable:**

**- 417 Expectation Failed:**

**- 5XX:** 服务器错误

**- 500 Internal Server Error:**

**- 501 Not Implemented:**

**- 502 Bad Gateway:**

**- 503 Service Unavailable:**

**- 504 Gateway Timeout:**

**- 505 HTTP Version Not Supported:**

**$CSS**部分

**CSS**选择器有哪些

**- \*\***通用选择器：选择所有元素，不参与计算优先级**\***，兼容性**IE6+**

**- #X id**选择器：选择**id**值为**X**的元素，兼容性：**IE6+**

**- .X** 类选择器：选择**class**包含**X**的元素，兼容性：**IE6+**

**- X Y**后代选择器：选择满足**X**选择器的后代节点中满足**Y**选择器的元素，兼容性：**IE6+**

**- X** 元素选择器：选择标所有签为**X**的元素，兼容性：**IE6+**

**- :link**，：**visited**，：**focus**，：**hover**，：**active**链接状态：选择特定状态的链接元素，顺序**LoVe HAte**，兼容性**: IE4+**

**- X + Y**直接兄弟选择器：在**X**之后第一个兄弟节点中选择满足**Y**选择器的元素，兼容性： **IE7+**

**- X > Y**子选择器：选择**X**的子元素中满足**Y**选择器的元素，兼容性： **IE7+**

**- X ~ Y**兄弟：选择**X**之后所有兄弟节点中满足**Y**选择器的元素，兼容性： **IE7+**

**- [attr]**：选择所有设置了**attr**属性的元素，兼容性**IE7+**

**- [attr=value]**：选择属性值刚好为**value**的元素

**- [attr~=value]**：选择属性值为空白符分隔，其中一个的值刚好是**value**的元素

**- [attr|=value]**：选择属性值刚好为**value**或者**value-**开头的元素

**- [attr^=value]**：选择属性值以**value**开头的元素

**- [attr$=value]**：选择属性值以**value**结尾的元素

**- [attr\*=value]**：选择属性值中包含**value**的元素

**- [:checked]**：选择单选框，复选框，下拉框中选中状态下的元素，兼容性：**IE9+**

**- X:after, X::after**：**after**伪元素，选择元素虚拟子元素（元素的最后一个子元素），**CSS3**中**::**表示伪元素。兼容性**:after**为**IE8+**，**::after**为**IE9+**

**- :hover**：鼠标移入状态的元素，兼容性**a**标签**IE4+**，所有元素**IE7+**

**- :not(selector)**：选择不符合**selector**的元素。不参与计算优先级，兼容性：**IE9+**

**- ::first-letter**：伪元素，选择块元素第一行的第一个字母，兼容性**IE5.5+**

**- ::first-line**：伪元素，选择块元素的第一行，兼容性**IE5.5+**

**- :nth-child(an + b)**：伪类，选择前面有**an + b - 1**个兄弟节点的元素，其中**n >= 0**，兼容性**IE9+**

**- :nth-last-child(an + b)**：伪类，选择后面有**an + b - 1**个兄弟节点的元素其中**n >= 0**，兼容性**IE9+**

**- X:nth-of-type(an+b)**：伪类，**X**为选择器，解析得到元素标签，选择前面有**an + b - 1**个相同标签兄弟节点的元素。兼容性**IE9+**

**- X:nth-last-of-type(an+b)**：伪类，**X**为选择器，解析得到元素标签，选择后面有**an+b-1**个相同标签兄弟节点的元素。兼容性**IE9+**

**- X:first-child**：伪类，选择满足**X**选择器的元素，且这个元素是其父节点的第一个子元素。兼容性**IE7+**

**- X:last-child**：伪类，选择满足**X**选择器的元素，且这个元素是其父节点的最后一个子元素。兼容性**IE9+**

**- X:only-child**：伪类，选择满足**X**选择器的元素，且这个元素是其父元素的唯一子元素。兼容性**IE9+**

**- X:only-of-type**：伪类，选择**X**选择的元素，解析得到元素标签，如果该元素没有相同类型的兄弟节点时选中它。兼容性**IE9+**

**- X:first-of-type**：伪类，选择**X**选择的元素，解析得到元素标签，如果该元素是此此类型元素的第一个兄弟。选中它。兼容性**IE9+**

**css sprite**是什么**,**有什么优缺点

概念：将多个小图片拼接到一个图片中。通过**background-position**和元素尺寸调节需要显示的背景图案。

优点：

**-** 减少**HTTP**请求数，极大地提高页面加载速度

**-** 增加图片信息重复度，提高压缩比，减少图片大小

**-** 更换风格方便，只需在一张或几张图片上修改颜色或样式即可实现

缺点：

**-** 图片合并麻烦

**-** 维护麻烦，修改一个图片可能需要从新布局整个图片，样式

**display: none;**与**visibility: hidden;**的区别

联系：它们都能让元素不可见

区别：

**- display:none;**会让元素完全从渲染树中消失，渲染的时候不占据任何空间；**visibility: hidden;**不会让元素从渲染树消失，渲染师元素继续占据空间，只是内容不可见

**- display: none;**是非继承属性，子孙节点消失由于元素从渲染树消失造成，通过修改子孙节点属性无法显示；**visibility: hidden;**是继承属性，子孙节点消失由于继承了**hidden**，通过设置**visibility: visible;**可以让子孙节点显式

**-** 修改常规流中元素的**display**通常会造成文档重排。修改**visibility**属性只会造成本元素的重绘。

**-** 读屏器不会读取**display: none;**元素内容；会读取**visibility: hidden;**元素内容

**css hack**原理及常用**hack**

原理：利用不同浏览器对**CSS**的支持和解析结果不一样编写针对特定浏览器样式。常见的**hack**有**1**）属性**hack**。**2**）选择器**hack**。**3**）**IE**条件注释

**- IE**条件注释：适用于**[IE5, IE9]**常见格式如下

**<!--[if IE 6]>**

**Special instructions for IE 6 here**

**<![endif]-->**

**-** 选择器**hack**：不同浏览器对选择器的支持不一样

**/\*\*\*\*\* Selector Hacks \*\*\*\*\*\*/**

**\* html #uno { color: red }**

**\*:first-child+html #dos { color: red }**

**html>body #tres { color: red }**

**html>body #cuatro { color: red }**

**html:first-child #cinco { color: red }**

**html[xmlns\*=""] body:last-child #seis { color: red }**

**body:nth-of-type(1) #siete { color: red }**

**body:first-of-type #ocho { color: red }**

**@media screen and (-webkit-min-device-pixel-ratio:0) {**

**#diez { color: red }**

**}**

**@media screen and (max-device-width: 480px) {**

**#veintiseis { color: red }**

**}**

**html[xmlns\*=""]:root #trece { color: red }**

**\*|html[xmlns\*=""] #catorce { color: red }**

**:root \*> #quince { color: red }**

**\*+html #dieciocho { color: red }**

**#veinticuatro, x:-moz-any-link { color: red }**

**#veinticinco, x:-moz-any-link, x:default { color: red }**

**-** 属性**hack**：不同浏览器解析**bug**或方法

**#once { \_color: blue }**

**#doce { \*color: blue; }**

**#diecisiete { color: blue }**

**#diecinueve { color: blue\9; }**

**#veinte { color/\*\\*\*/: blue\9; }**

**#veintesiete { color: blue !ie; }**

**specified value,computed value,used value**计算方法

**- specified value:** 计算方法如下：

**-** 如果样式表设置了一个值，使用这个值

**-** 如果没有设置值，这个属性是继承属性，从父元素继承

**-** 如果没设置，并且不是继承属性，使用**css**规范指定的初始值

**- computed value:** 以**specified value**根据规范定义的行为进行计算，通常将相对值计算为绝对值，例如**em**根据**font-size**进行计算。一些使用百分数并且需要布局来决定最终值的属性，如**width**，**margin**。百分数就直接作为**computed value**。**line-height**的无单位值也直接作为**computed value**。这些值将在计算**used value**时得到绝对值。**computed value**的主要作用是用于继承

**- used value**：属性计算后的最终值，对于大多数属性可以通过**window.getComputedStyle**获得，尺寸值单位为像素。以下属性依赖于布局，

**- background-position**

**- bottom, left, right, top**

**- height, width**

**- margin-bottom, margin-left, margin-right, margin-top**

**- min-height, min-width**

**- padding-bottom, padding-left, padding-right, padding-top**

**- text-indent**

**link**与**@import**的区别

**- link**是**HTML**方式， **@import**是**CSS**方式

**- link**最大限度支持并行下载，**@import**过多嵌套导致串行下载，出现**FOUC**

**- link**可以通过**rel="alternate stylesheet"**指定候选样式

**-** 浏览器对**link**支持早于**@import**，可以使用**@import**对老浏览器隐藏样式

**- @import**必须在样式规则之前，可以在**css**文件中引用其他文件

**-** 总体来说：**link**优于**@import**

**display: block;**和**display: inline;**的区别

**block**元素特点：

**1.**处于常规流中时，如果**width**没有设置，会自动填充满父容器 **2.**可以应用**margin/padding 3.**在没有设置高度的情况下会扩展高度以包含常规流中的子元素 **4.**处于常规流中时布局时在前后元素位置之间（独占一个水平空间） **5.**忽略**vertical-align**

**inline**元素特点

**1.**水平方向上根据**direction**依次布局 **2.**不会在元素前后进行换行 **3.**受**white-space**控制 **4.margin/padding**在竖直方向上无效，水平方向上有效 **5.width/height**属性对非替换行内元素无效，宽度由元素内容决定 **6.**非替换行内元素的行框高由**line-height**确定，替换行内元素的行框高由**height,margin,padding,border**决定 **6.**浮动或绝对定位时会转换为**block7.vertical-align**属性生效

**PNG,GIF,JPG**的区别及如何选

参考资料：选择正确的图片格式 **GIF:**

**- 8**位像素，**256**色

**-** 无损压缩

**-** 支持简单动画

**-** 支持**boolean**透明

**-** 适合简单动画

**JPEG**：

**-** 颜色限于**256**

**-** 有损压缩

**-** 可控制压缩质量

**-** 不支持透明

**-** 适合照片

**PNG**：

**-** 有**PNG8**和**truecolor PNG**

**- PNG8**类似**GIF**颜色上限为**256**，文件小，支持**alpha**透明度，无动画

**-** 适合图标、背景、按钮

**CSS**有哪些继承属性

**-** 关于文字排版的属性如：

**- font**

**- word-break**

**- letter-spacing**

**- text-align**

**- text-rendering**

**- word-spacing**

**- white-space**

**- text-indent**

**- text-transform**

**- text-shadow**

**- line-height**

**- color**

**- visibility**

**- cursor**

**IE6**浏览器有哪些常见的**bug,**缺陷或者与标准不一致的地方**,**如何解决

**- IE6**不支持**min-height**，解决办法使用**css hack**：

**.target {**

**min-height: 100px;**

**height: auto !important;**

**height: 100px; // IE6**下内容高度超过会自动扩展高度

**}**

**- ol**内**li**的序号全为**1**，不递增。解决方法：为**li**设置样式**display: list-item;**

**-** 未定位父元素**overflow: auto;**，包含**position: relative;**子元素，子元素高于父元素时会溢出。解决办法：**1**）子元素去掉**position: relative;; 2**）不能为子元素去掉定位时，父元素**position: relative;**

**<style type="text/css">**

**.outer {**

**width: 215px;**

**height: 100px;**

**border: 1px solid red;**

**overflow: auto;**

**position: relative;**

**}**

**.inner {**

**width: 100px;**

**height: 200px;**

**background-color: purple;**

**position: relative;**

**}**

**</style>**

**<div class="outer">**

**<div class="inner"></div>**

**</div>**

**- IE6**只支持**a**标签的**:hover**伪类，解决方法：使用**js**为元素监听**mouseenter**，**mouseleave**事件，添加类实现效果：

**<style type="text/css">**

**.p:hover,**

**.hover {**

**background: purple;**

**}**

**</style>**

**<p class="p" id="target">aaaa bbbbb<span>DDDDDDDDDDDd</span> aaaa lkjlkjdf j</p>**

**<script type="text/javascript">**

**function addClass(elem, cls) {**

**if (elem.className) {**

**elem.className += ' ' + cls;**

**} else {**

**elem.className = cls;**

**}**

**}**

**function removeClass(elem, cls) {**

**var className = ' ' + elem.className + ' ';**

**var reg = new RegExp(' +' + cls + ' +', 'g');**

**elem.className = className.replace(reg, ' ').replace(/^ +| +$/, '');**

**}**

**var target = document.getElementById('target');**

**if (target.attachEvent) {**

**target.attachEvent('onmouseenter', function () {**

**addClass(target, 'hover');**

**});**

**target.attachEvent('onmouseleave', function () {**

**removeClass(target, 'hover');**

**})**

**}**

**</script>**

**- IE5-8**不支持**opacity**，解决办法：

**.opacity {**

**opacity: 0.4**

**filter: alpha(opacity=60);**

**-ms-filter: "progid:DXImageTransform.Microsoft.Alpha(Opacity=60)";**

**}**

**- IE6**在设置**height**小于**font-size**时高度值为**font-size**，解决办法：**font-size: 0;**

**- IE6**不支持**PNG**透明背景，解决办法**: IE6**下使用**gif**图片

**- IE6-7**不支持**display: inline-block**解决办法：设置**inline**并触发**hasLayout**

**display: inline-block;**

**\*display: inline;**

**\*zoom: 1;**

**- IE6**下浮动元素在浮动方向上与父元素边界接触元素的外边距会加倍。解决办法： **1**）使用**padding**控制间距。 **2**）浮动元素**display: inline;**这样解决问题且无任何副作用：**css**标准规定浮动元素**display:inline**会自动调整为**block**

**-** 通过为块级元素设置宽度和左右**margin**为**auto**时，**IE6**不能实现水平居中，解决方法：为父元素设置**text-align: center;**

容器包含若干浮动元素时如何清理**(**包含**)**浮动

**-** 容器元素闭合标签前添加额外元素并设置**clear: both**

**-** 父元素触发块级格式化上下文**(**见块级可视化上下文部分**)**

**-** 设置容器元素伪元素进行清理推荐的清理浮动方法

**/\*\***

**\*** 在标准浏览器下使用

**\* 1 content**内容为空格用于修复**opera**下文档中出现

**\* contenteditable**属性时在清理浮动元素上下的空白

**\* 2** 使用**display**使用**table**而不是**block**：可以防止容器和

**\*** 子元素**top-margin**折叠**,**这样能使清理效果与**BFC**，**IE6/7**

**\* zoom: 1;**一致

**\*\*/**

**.clearfix:before,**

**.clearfix:after {**

**content: " ";**

**display: table;**

**}**

**.clearfix:after {**

**clear: both;**

**}**

**/\*\***

**\* IE 6/7**下使用

**\*** 通过触发**hasLayout**实现包含浮动

**\*\*/**

**.clearfix {**

**\*zoom: 1;**

**}**

什么是**FOUC?**如何避免

**Flash Of Unstyled Content**：用户定义样式表加载之前浏览器使用默认样式显示文档，用户样式加载渲染之后再从新显示文档，造成页面闪烁。解决方法：把样式表放到文档的**head**

如何创建块级格式化上下文**(block formatting context),BFC**有什么用

创建规则：

**-** 根元素

**-** 浮动元素（**float**不是**none**）

**-** 绝对定位元素（**position**取值为**absolute**或**fixed**）

**- display**取值为**inline-block,table-cell, table-caption,flex, inline-flex**之一的元素

**- overflow**不是**visible**的元素

作用：

**-** 可以包含浮动元素

**-** 不被浮动元素覆盖

**-** 阻止父子元素的**margin**折叠

**display,float,position**的关系

**-** 如果**display**为**none**，那么**position**和**float**都不起作用，这种情况下元素不产生框

**-** 否则，如果**position**值为**absolute**或者**fixed**，框就是绝对定位的，**float**的计算值为**none**，**display**根据下面的表格进行调整。

**-** 否则，如果**float**不是**none**，框是浮动的，**display**根据下表进行调整

**-** 否则，如果元素是根元素，**display**根据下表进行调整

**-** 其他情况下**display**的值为指定值总结起来：绝对定位、浮动、根元素都需要调整**display**

外边距折叠**(collapsing margins)**

毗邻的两个或多个**margin**会合并成一个**margin**，叫做外边距折叠。规则如下：

**-** 两个或多个毗邻的普通流中的块元素垂直方向上的**margin**会折叠

**-** 浮动元素**/inline-block**元素**/**绝对定位元素的**margin**不会和垂直方向上的其他元素的**margin**折叠

**-** 创建了块级格式化上下文的元素，不会和它的子元素发生**margin**折叠

**-** 元素自身的**margin-bottom**和**margin-top**相邻时也会折叠

如何确定一个元素的包含块**(containing block)**

**-** 根元素的包含块叫做初始包含块，在连续媒体中他的尺寸与**viewport**相同并且**anchored at the canvas origin**；对于**paged media**，它的尺寸等于**page area**。初始包含块的**direction**属性与根元素相同。

**- position**为**relative**或者**static**的元素，它的包含块由最近的块级（**display**为**block,list-item, table**）祖先元素的内容框组成

**-** 如果元素**position**为**fixed**。对于连续媒体，它的包含块为**viewport**；对于**paged media**，包含块为**page area**

**-** 如果元素**position**为**absolute**，它的包含块由祖先元素中最近一个**position**为**relative,absolute**或者**fixed**的元素产生，规则如下：

**-** 如果祖先元素为行内元素，**the containing block is the bounding box around the padding boxes of the first and the last inline boxes generated for that element.**

**-** 其他情况下包含块由祖先节点的**padding edge**组成

**-** 如果找不到定位的祖先元素，包含块为初始包含块

**stacking context,**布局规则

**z**轴上的默认层叠顺序如下（从下到上）：

**-** 根元素的边界和背景

**-** 常规流中的元素按照**html**中顺序

**-** 浮动块

**- positioned**元素按照**html**中出现顺序

如何创建**stacking context**：

**-** 根元素

**- z-index**不为**auto**的定位元素

**- a flex item with a z-index value other than 'auto'**

**- opacity**小于**1**的元素

**-** 在移动端**webkit**和**chrome22+**，**z-index**为**auto**，**position: fixed**也将创建新的**stacking context**

如何水平居中一个元素

**-** 如果需要居中的元素为常规流中**inline**元素，为父元素设置**text-align: center;**即可实现

**-** 如果需要居中的元素为常规流中**block**元素，**1**）为元素设置宽度，**2**）设置左右**margin**为**auto**。**3**）**IE6**下需在父元素上设置**text-align: center;,**再给子元素恢复需要的值

**<body>**

**<div class="content">**

**aaaaaa aaaaaa a a a a a a a a**

**</div>**

**</body>**

**<style>**

**body {**

**background: #DDD;**

**text-align: center;**

**}**

**.content {**

**width: 500px;**

**text-align: left;**

**margin: 0 auto;**

**background: purple;**

**}**

**</style>**

**-** 如果需要居中的元素为浮动元素，**1**）为元素设置宽度，**2**）**position: relative;**，**3**）浮动方向偏移量（**left**或者**right**）设置为**50%**，**4**）浮动方向上的**margin**设置为元素宽度一半乘以**-1**

**<body>**

**<div class="content">**

**aaaaaa aaaaaa a a a a a a a a**

**</div>**

**</body>**

**<style>**

**body {**

**background: #DDD;**

**}**

**.content {**

**width: 500px;**

**float: left;**

**position: relative;**

**left: 50%;**

**margin-left: -250px;**

**background-color: purple;**

**}**

**</style>**

**-** 如果需要居中的元素为绝对定位元素，**1**）为元素设置宽度，**2**）偏移量设置为**50%**，**3**）偏移方向外边距设置为元素宽度一半乘以**-1**

**<body>**

**<div class="content">**

**aaaaaa aaaaaa a a a a a a a a**

**</div>**

**</body>**

**<style>**

**body {**

**background: #DDD;**

**position: relative;**

**}**

**.content {**

**width: 800px;**

**position: absolute;**

**left: 50%;**

**margin-left: -400px;**

**background-color: purple;**

**}**

**</style>**

**-** 如果需要居中的元素为绝对定位元素，**1**）为元素设置宽度，**2**）设置左右偏移量都为**0,3**）设置左右外边距都为**auto**

**<body>**

**<div class="content">**

**aaaaaa aaaaaa a a a a a a a a**

**</div>**

**</body>**

**<style>**

**body {**

**background: #DDD;**

**position: relative;**

**}**

**.content {**

**width: 800px;**

**position: absolute;**

**margin: 0 auto;**

**left: 0;**

**right: 0;**

**background-color: purple;**

**}**

**</style>**

如何竖直居中一个元素

参考资料：**6 Methods For Vertical Centering With CSS**。盘点**8**种**CSS**实现垂直居中

**-** 需要居中元素为单行文本，为包含文本的元素设置大于**font-size**的**line-height**：

**<p class="text">center text</p>**

**<style>**

**.text {**

**line-height: 200px;**

**}**

**</style>**

**$javascript**概念部分

**DOM**元素**e**的**e.getAttribute(propName)**和**e.propName**有什么区别和联系

**- e.getAttribute()**，是标准**DOM**操作文档元素属性的方法，具有通用性可在任意文档上使用，返回元素在源文件中设置的属性

**- e.propName**通常是在**HTML**文档中访问特定元素的特性，浏览器解析元素后生成对应对象（如**a**标签生成**HTMLAnchorElement**），这些对象的特性会根据特定规则结合属性设置得到，对于没有对应特性的属性，只能使用**getAttribute**进行访问

**- e.getAttribute()**返回值是源文件中设置的值，类型是字符串或者**null**（有的实现返回**""**）

**- e.propName**返回值可能是字符串、布尔值、对象、**undefined**等

**-** 大部分**attribute**与**property**是一一对应关系，修改其中一个会影响另一个，如**id**，**title**等属性

**-** 一些布尔属性**<input hidden/>**的检测设置需要**hasAttribute**和**removeAttribute**来完成，或者设置对应**property**

**-** 像**<a href="../index.html">link</a>**中**href**属性，转换成**property**的时候需要通过转换得到完整**URL**

**-** 一些**attribute**和**property**不是一一对应如：**form**控件中**<input value="hello"/>**对应的是**defaultValue**，修改或设置**value property**修改的是控件当前值，**setAttribute**修改**value**属性不会改变**value property**

**offsetWidth/offsetHeight,clientWidth/clientHeight**与**scrollWidth/scrollHeight**的区别

**- offsetWidth/offsetHeight**返回值包含**content + padding + border**，效果与**e.getBoundingClientRect()**相同

**- clientWidth/clientHeight**返回值只包含**content + padding**，如果有滚动条，也不包含滚动条

**- scrollWidth/scrollHeight**返回值包含**content + padding +** 溢出内容的尺寸

**Measuring Element Dimension and Location with CSSOM in Windows Internet Explorer 9**

**XMLHttpRequest**通用属性和方法

**- readyState:**表示请求状态的整数，取值：

**- UNSENT**（**0**）：对象已创建

**- OPENED**（**1**）：**open()**成功调用，在这个状态下，可以为**xhr**设置请求头，或者使用**send()**发送请求

**- HEADERS\_RECEIVED(2)**：所有重定向已经自动完成访问，并且最终响应的**HTTP**头已经收到

**- LOADING(3)**：响应体正在接收

**- DONE(4)**：数据传输完成或者传输产生错误

**- onreadystatechange**：**readyState**改变时调用的函数

**- status**：服务器返回的**HTTP**状态码（如，**200**， **404**）

**- statusText:**服务器返回的**HTTP**状态信息（如，**OK**，**No Content**）

**- responseText:**作为字符串形式的来自服务器的完整响应

**- responseXML: Document**对象，表示服务器的响应解析成的**XML**文档

**- abort():**取消异步**HTTP**请求

**- getAllResponseHeaders():** 返回一个字符串，包含响应中服务器发送的全部**HTTP**报头。每个报头都是一个用冒号分隔开的名**/**值对，并且使用一个回车**/**换行来分隔报头行

**- getResponseHeader(headerName):**返回**headName**对应的报头值

**- open(method, url, asynchronous [, user, password]):**初始化准备发送到服务器上的请求。**method**是**HTTP**方法，不区分大小写；**url**是请求发送的相对或绝对**URL**；**asynchronous**表示请求是否异步；**user**和**password**提供身份验证

**- setRequestHeader(name, value):**设置**HTTP**报头

**- send(body):**对服务器请求进行初始化。参数**body**包含请求的主体部分，对于**POST**请求为键值对字符串；对于**GET**请求，为**null**

**focus/blur**与**focusin/focusout**的区别与联系

**- focus/blur**不冒泡，**focusin/focusout**冒泡

**- focus/blur**兼容性好，**focusin/focusout**在除**FireFox**外的浏览器下都保持良好兼容性，如需使用事件托管，可考虑在**FireFox**下使用事件捕获**elem.addEventListener('focus', handler, true)**

**-** 可获得焦点的元素：

**- window**

**-** 链接被点击或键盘操作

**-** 表单空间被点击或键盘操作

**-** 设置**tabindex**属性的元素被点击或键盘操作

**mouseover/mouseout**与**mouseenter/mouseleave**的区别与联系

**- mouseover/mouseout**是标准事件，所有浏览器都支持；**mouseenter/mouseleave**是**IE5.5**引入的特有事件后来被**DOM3**标准采纳，现代标准浏览器也支持

**- mouseover/mouseout**是冒泡事件；**mouseenter/mouseleave**不冒泡。需要为多个元素监听鼠标移入**/**出事件时，推荐**mouseover/mouseout**托管，提高性能

**-** 标准事件模型中**event.target**表示发生移入**/**出的元素**,vent.relatedTarget**对应移出**/**如元素；在老**IE**中**event.srcElement**表示发生移入**/**出的元素，**event.toElement**表示移出的目标元素，**event.fromElement**表示移入时的来源元素

例子：鼠标从**div#target**元素移出时进行处理，判断逻辑如下：

**<div id="target"><span>test</span></div>**

**<script type="text/javascript">**

**var target = document.getElementById('target');**

**if (target.addEventListener) {**

**target.addEventListener('mouseout', mouseoutHandler, false);**

**} else if (target.attachEvent) {**

**target.attachEvent('onmouseout', mouseoutHandler);**

**}**

**function mouseoutHandler(e) {**

**e = e || window.event;**

**var target = e.target || e.srcElement;**

**//** 判断移出鼠标的元素是否为目标元素

**if (target.id !== 'target') {**

**return;**

**}**

**//** 判断鼠标是移出元素还是移到子元素

**var relatedTarget = event.relatedTarget || e.toElement;**

**while (relatedTarget !== target**

**&& relatedTarget.nodeName.toUpperCase() !== 'BODY') {**

**relatedTarget = relatedTarget.parentNode;**

**}**

**//** 如果相等，说明鼠标在元素内部移动

**if (relatedTarget === target) {**

**return;**

**}**

**//** 执行需要操作

**//alert('**鼠标移出**');**

**}**

**</script>**

**sessionStorage,localStorage,cookie**区别

**-** 都会在浏览器端保存，有大小限制，同源限制

**- cookie**会在请求时发送到服务器，作为会话标识，服务器可修改**cookie**；**web storage**不会发送到服务器

**- cookie**有**path**概念，子路径可以访问父路径**cookie**，父路径不能访问子路径**cookie**

**-** 有效期：**cookie**在设置的有效期内有效，默认为浏览器关闭；**sessionStorage**在窗口关闭前有效，**localStorage**长期有效，直到用户删除

**-** 共享：**sessionStorage**不能共享，**localStorage**在同源文档之间共享，**cookie**在同源且符合**path**规则的文档之间共享

**- localStorage**的修改会促发其他文档窗口的**update**事件

**- cookie**有**secure**属性要求**HTTPS**传输

**-** 浏览器不能保存超过**300**个**cookie**，单个服务器不能超过**20**个，每个**cookie**不能超过**4k**。**web storage**大小支持能达到**5M**

**javascript**跨域通信

同源：两个文档同源需满足

**-** 协议相同

**-** 域名相同

**-** 端口相同

跨域通信：**js**进行**DOM**操作、通信时如果目标与当前窗口不满足同源条件，浏览器为了安全会阻止跨域操作。跨域通信通常有以下方法

**-** 如果是**log**之类的简单单项通信，新建**<img>,<script>,<link>,<iframe>**元素，通过**src**，**href**属性设置为目标**url**。实现跨域请求

**-** 如果请求**json**数据，使用**<script>**进行**jsonp**请求

**-** 现代浏览器中多窗口通信使用**HTML5**规范的**targetWindow.postMessage(data, origin);**其中**data**是需要发送的对象，**origin**是目标窗口的**origin**。**window.addEventListener('message', handler, false);handler**的**event.data**是**postMessage**发送来的数据，**event.origin**是发送窗口的**origin**，**event.source**是发送消息的窗口引用

**-** 内部服务器代理请求跨域**url**，然后返回数据

**-** 跨域请求数据，现代浏览器可使用**HTML5**规范的**CORS**功能，只要目标服务器返回**HTTP**头部**Access-Control-Allow-Origin: \***即可像普通**ajax**一样访问跨域资源

**javascript**有哪几种数据类型

六种基本数据类型

**- undefined**

**- null**

**- string**

**- boolean**

**- number**

**- symbol(ES6)**

一种引用类型

**- Object**

什么闭包**,**闭包有什么用

闭包是在某个作用域内定义的函数，它可以访问这个作用域内的所有变量。闭包作用域链通常包括三个部分：

**-** 函数本身作用域。

**-** 闭包定义时的作用域。

**-** 全局作用域。

闭包常见用途：

**-** 创建特权方法用于访问控制

**-** 事件处理程序及回调

**javascript**有哪几种方法定义函数

**-** 函数声明表达式

**- function**操作符

**- Function** 构造函数

**- ES6:arrow function**

重要参考资料：**MDN:Functions\_and\_function\_scope**

应用程序存储和离线**web**应用

**HTML5**新增应用程序缓存，允许**web**应用将应用程序自身保存到用户浏览器中，用户离线状态也能访问。 **1.**为**html**元素设置**manifest**属性**:<html manifest="myapp.appcache">**，其中后缀名只是一个约定，真正识别方式是通过**text/cache-manifest**作为**MIME**类型。所以需要配置服务器保证设置正确 **2.manifest**文件首行为**CACHE MANIFEST**，其余就是要缓存的**URL**列表，每个一行，相对路径都相对于**manifest**文件的**url**。注释以**#**开头 **3.url**分为三种类型：**CACHE:**为默认类型。**NETWORK**：表示资源从不缓存。 **FALLBACK:**每行包含两个**url**，第二个**URL**是指需要加载和存储在缓存中的资源，第一个**URL**是一个前缀。任何匹配该前缀的**URL**都不会缓存，如果从网络中载入这样的**URL**失败的话，就会用第二个**URL**指定的缓存资源来替代。以下是一个文件例子：

**CACHE MANIFEST**

**CACHE:**

**myapp.html**

**myapp.css**

**myapp.js**

**FALLBACK:**

**videos/ offline\_help.html**

**NETWORK:**

**cgi/**

客户端存储**localStorage**和**sessionStorage**

**- localStorage**有效期为永久，**sessionStorage**有效期为顶层窗口关闭前

**-** 同源文档可以读取并修改**localStorage**数据，**sessionStorage**只允许同一个窗口下的文档访问，如通过**iframe**引入的同源文档。

**- Storage**对象通常被当做普通**javascript**对象使用：通过设置属性来存取字符串值，也可以通过**setItem(key, value)**设置，**getItem(key)**读取，**removeItem(key)**删除，**clear()**删除所有数据，**length**表示已存储的数据项数目，**key(index)**返回对应索引的**key**

**localStorage.setItem('x', 1); // storge x->1**

**localStorage.getItem('x); // return value of x**

**//** 枚举所有存储的键值对

**for (var i = 0, len = localStorage.length; i < len; ++i ) {**

**var name = localStorage.key(i);**

**var value = localStorage.getItem(name);**

**}**

**localStorage.removeItem('x'); // remove x**

**localStorage.clear(); // remove all data**

**cookie**及其操作

**- cookie**是**web**浏览器存储的少量数据，最早设计为服务器端使用，作为**HTTP**协议的扩展实现。**cookie**数据会自动在浏览器和服务器之间传输。

**-** 通过读写**cookie**检测是否支持

**- cookie**属性有名，值，**max-age**，**path, domain**，**secure**；

**- cookie**默认有效期为浏览器会话，一旦用户关闭浏览器，数据就丢失，通过设置**max-age=seconds**属性告诉浏览器**cookie**有效期

**- cookie**作用域通过文档源和文档路径来确定，通过**path**和**domain**进行配置，**web**页面同目录或子目录文档都可访问

**-** 通过**cookie**保存数据的方法为：为**document.cookie**设置一个符合目标的字符串如下

**-** 读取**document.cookie**获得**'; '**分隔的字符串，**key=value,**解析得到结果

**document.cookie = 'name=qiu; max-age=9999; path=/; domain=domain; secure';**

**document.cookie = 'name=aaa; path=/; domain=domain; secure';**

**//** 要改变**cookie**的值，需要使用相同的名字、路径和域，新的值

**//** 来设置**cookie**，同样的方法可以用来改变有效期

**//** 设置**max-age**为**0**可以删除指定**cookie**

**//**读取**cookie**，访问**document.cookie**返回键值对组成的字符串，

**//**不同键值对之间用**'; '**分隔。通过解析获得需要的值

**cookieUtil.js**：自己写的**cookie**操作工具

**javascript**有哪些方法定义对象

**-** 对象字面量： **var obj = {};**

**-** 构造函数： **var obj = new Object();**

**- Object.create(): var obj = Object.create(Object.prototype);**

**===**运算符判断相等的流程是怎样的

**-** 如果两个值不是相同类型，它们不相等

**-** 如果两个值都是**null**或者都是**undefined**，它们相等

**-** 如果两个值都是布尔类型**true**或者都是**false**，它们相等

**-** 如果其中有一个是**NaN**，它们不相等

**-** 如果都是数值型并且数值相等，他们相等， **-0**等于**0**

**-** 如果他们都是字符串并且在相同位置包含相同的**16**位值，他它们相等；如果在长度或者内容上不等，它们不相等；两个字符串显示结果相同但是编码不同**==**和**===**都认为他们不相等

**-** 如果他们指向相同对象、数组、函数，它们相等；如果指向不同对象，他们不相等

**==**运算符判断相等的流程是怎样的

**-** 如果两个值类型相同，按照**===**比较方法进行比较

**-** 如果类型不同，使用如下规则进行比较

**-** 如果其中一个值是**null**，另一个是**undefined**，它们相等

**-** 如果一个值是数字另一个是字符串，将字符串转换为数字进行比较

**-** 如果有布尔类型，将**true**转换为**1**，**false**转换为**0**，然后用**==**规则继续比较

**-** 如果一个值是对象，另一个是数字或字符串，将对象转换为原始值然后用**==**规则继续比较

**-** 其他所有情况都认为不相等

对象到字符串的转换步骤

**-** 如果对象有**toString()**方法，**javascript**调用它。如果返回一个原始值（**primitive value**如：**string number boolean**）**,**将这个值转换为字符串作为结果

**-** 如果对象没有**toString()**方法或者返回值不是原始值，**javascript**寻找对象的**valueOf()**方法，如果存在就调用它，返回结果是原始值则转为字符串作为结果

**-** 否则，**javascript**不能从**toString()**或者**valueOf()**获得一个原始值，此时**throws a TypeError**

对象到数字的转换步骤

**1.** 如果对象有**valueOf()**方法并且返回元素值，**javascript**将返回值转换为数字作为结果

**2.** 否则，如果对象有**toString()**并且返回原始值，**javascript**将返回结果转换为数字作为结果

**3.** 否则，**throws a TypeError**

**<,>,<=,>=**的比较规则

所有比较运算符都支持任意类型，但是比较只支持数字和字符串，所以需要执行必要的转换然后进行比较，转换规则如下**: 1.** 如果操作数是对象，转换为原始值：如果**valueOf**方法返回原始值，则使用这个值，否则使用**toString**方法的结果，如果转换失败则报错 **2.** 经过必要的对象到原始值的转换后，如果两个操作数都是字符串，按照字母顺序进行比较（他们的**16**位**unicode**值的大小） **3.** 否则，如果有一个操作数不是字符串，将两个操作数转换为数字进行比较

**+**运算符工作流程

**-** 如果有操作数是对象，转换为原始值

**-** 此时如果有一个操作数是字符串，其他的操作数都转换为字符串并执行连接

**-** 否则：所有操作数都转换为数字并执行加法

函数内部**arguments**变量有哪些特性**,**有哪些属性**,**如何将它转换为数组

**- arguments**所有函数中都包含的一个局部变量，是一个类数组对象，对应函数调用时的实参。如果函数定义同名参数会在调用时覆盖默认对象

**- arguments[index]**分别对应函数调用时的实参，并且通过**arguments**修改实参时会同时修改实参

**- arguments.length**为实参的个数（**Function.length**表示形参长度）

**- arguments.callee**为当前正在执行的函数本身，使用这个属性进行递归调用时需注意**this**的变化

**- arguments.caller**为调用当前函数的函数（已被遗弃）

**-** 转换为数组：**var args = Array.prototype.slice.call(arguments, 0);**

**DOM**事件模型是如何的**,**编写一个**EventUtil**工具类实现事件管理兼容

**- DOM**事件包含捕获（**capture**）和冒泡（**bubble**）两个阶段：捕获阶段事件从**window**开始触发事件然后通过祖先节点一次传递到触发事件的**DOM**元素上；冒泡阶段事件从初始元素依次向祖先节点传递直到**window**

**-** 标准事件监听**elem.addEventListener(type, handler, capture)/elem.removeEventListener(type, handler, capture)**：**handler**接收保存事件信息的**event**对象作为参数，**event.target**为触发事件的对象，**handler**调用上下文**this**为绑定监听器的对象，**event.preventDefault()**取消事件默认行为，**event.stopPropagation()/event.stopImmediatePropagation()**取消事件传递

**-** 老版本**IE**事件监听**elem.attachEvent('on'+type, handler)/elem.detachEvent('on'+type, handler)**：**handler**不接收**event**作为参数，事件信息保存在**window.event**中，触发事件的对象为**event.srcElement**，**handler**执行上下文**this**为**window**使用闭包中调用**handler.call(elem, event)**可模仿标准模型，然后返回闭包，保证了监听器的移除。**event.returnValue**为**false**时取消事件默认行为，**event.cancleBubble**为**true**时取消时间传播

**-** 通常利用事件冒泡机制托管事件处理程序提高程序性能。

**/\*\***

**\*** 跨浏览器事件处理工具。只支持冒泡。不支持捕获

**\* @author (qiu\_deqing@126.com)**

**\*/**

**var EventUtil = {**

**getEvent: function (event) {**

**return event || window.event;**

**},**

**getTarget: function (event) {**

**return event.target || event.srcElement;**

**},**

**//** 返回注册成功的监听器，**IE**中需要使用返回值来移除监听器

**on: function (elem, type, handler) {**

**if (elem.addEventListener) {**

**elem.addEventListener(type, handler, false);**

**return handler;**

**} else if (elem.attachEvent) {**

**var wrapper = function () {**

**var event = window.event;**

**event.target = event.srcElement;**

**handler.call(elem, event);**

**};**

**elem.attachEvent('on' + type, wrapper);**

**return wrapper;**

**}**

**},**

**off: function (elem, type, handler) {**

**if (elem.removeEventListener) {**

**elem.removeEventListener(type, handler, false);**

**} else if (elem.detachEvent) {**

**elem.detachEvent('on' + type, handler);**

**}**

**},**

**preventDefault: function (event) {**

**if (event.preventDefault) {**

**event.preventDefault();**

**} else if ('returnValue' in event) {**

**event.returnValue = false;**

**}**

**},**

**stopPropagation: function (event) {**

**if (event.stopPropagation) {**

**event.stopPropagation();**

**} else if ('cancelBubble' in event) {**

**event.cancelBubble = true;**

**}**

**},**

**/\*\***

**\* keypress**事件跨浏览器获取输入字符

**\*** 某些浏览器在一些特殊键上也触发**keypress**，此时返回**null**

**\*\*/**

**getChar: function (event) {**

**if (event.which == null) {**

**return String.fromCharCode(event.keyCode); // IE**

**}**

**else if (event.which != 0 && event.charCode != 0) {**

**return String.fromCharCode(event.which); // the rest**

**}**

**else {**

**return null; // special key**

**}**

**}**

**};**

评价一下三种方法实现继承的优缺点**,**并改进

**function Shape() {}**

**function Rect() {}**

**//** 方法**1**

**Rect.prototype = new Shape();**

**//** 方法**2**

**Rect.prototype = Shape.prototype;**

**//** 方法**3**

**Rect.prototype = Object.create(Shape.prototype);**

**Rect.prototype.area = function () {**

**// do something**

**};**

方法**1**：

**-** 优点：正确设置原型链实现继承

**-** 优点：父类实例属性得到继承，原型链查找效率提高，也能为一些属性提供合理的默认值

**-** 缺点：父类实例属性为引用类型时，不恰当地修改会导致所有子类被修改

**-** 缺点：创建父类实例作为子类原型时，可能无法确定构造函数需要的合理参数，这样提供的参数继承给子类没有实际意义，当子类需要这些参数时应该在构造函数中进行初始化和设置

**-** 总结：继承应该是继承方法而不是属性，为子类设置父类实例属性应该是通过在子类构造函数中调用父类构造函数进行初始化

方法**2**：

**-** 优点：正确设置原型链实现继承

**-** 缺点：父类构造函数原型与子类相同。修改子类原型添加方法会修改父类

方法**3**：

**-** 优点：正确设置原型链且避免方法**1.2**中的缺点

**-** 缺点：**ES5**方法需要注意兼容性

改进：

**-** 所有三种方法应该在子类构造函数中调用父类构造函数实现实例属性初始化

**function Rect() {**

**Shape.call(this);**

**}**

**-** 用新创建的对象替代子类默认原型，设置**Rect.prototype.constructor = Rect;**保证一致性

**-** 第三种方法的**polyfill**：

**function create(obj) {**

**if (Object.create) {**

**return Object.create(obj);**

**}**

**function f() {};**

**f.prototype = obj;**

**return new f();**

**}**

**$javascript**编程部分

请用原生**js**实现一个函数**,**给页面制定的任意一个元素添加一个透明遮罩**(**透明度可变**,**默认**0.2),**使这个区域点击无效**,**要求兼容**IE8+**及各主流浏览器**,**遮罩层效果如下图所示**:**

**<style>**

**#target {**

**width: 200px;**

**height: 300px;**

**margin: 40px;**

**background-color: tomato;**

**}**

**</style>**

**<div id="target"></div>**

**<script>**

**function addMask(elem, opacity) {**

**opacity = opacity || 0.2;**

**var rect = elem.getBoundingClientRect();**

**var style = getComputedStyle(elem, null);**

**var mask = document.createElement('div');**

**mask.style.position = 'absolute';**

**var marginLeft = parseFloat(style.marginLeft);**

**mask.style.left = (elem.offsetLeft - marginLeft) + 'px';**

**var marginTop = parseFloat(style.marginTop);**

**mask.style.top = (elem.offsetTop - marginTop) + 'px';**

**mask.style.zIndex = 9999;**

**mask.style.opacity = '' + opacity;**

**mask.style.backgroundColor = '#000';**

**mask.style.width = (parseFloat(style.marginLeft) +**

**parseFloat(style.marginRight) + rect.width) + 'px';**

**mask.style.height = (parseFloat(style.marginTop) +**

**parseFloat(style.marginBottom) + rect.height) + 'px';**

**elem.parentNode.appendChild(mask);**

**}**

**var target = document.getElementById('target');**

**addMask(target);**

**target.addEventListener('click', function () {**

**console.log('click');**

**}, false);**

**</script>**

请用代码写出**(**今天是星期**x)**其中**x**表示当天是星期几**,**如果当天是星期一**,**输出应该是**"**今天是星期一**"**

**var days = ['**日**','**一**','**二**','**三**','**四**','**五**','**六**'];**

**var date = new Date();**

**console.log('**今天是星期**' + days[date.getDay()]);**

下面这段代码想要循环延时输出结果**0 1 2 3 4,**请问输出结果是否正确**,**如果不正确**,**请说明为什么**,**并修改循环内的代码使其输出正确结果

**for (var i = 0; i < 5; ++i) {**

**setTimeout(function () {**

**console.log(i + ' ');**

**}, 100);**

**}**

不能输出正确结果，因为循环中**setTimeout**接受的参数函数通过闭包访问变量**i**。**javascript**运行环境为单线程，**setTimeout**注册的函数需要等待线程空闲才能执行，此时**for**循环已经结束，**i**值为**5.**五个定时输出都是**5** 修改方法：将**setTimeout**放在函数立即调用表达式中，将**i**值作为参数传递给包裹函数，创建新闭包

**for (var i = 0; i < 5; ++i) {**

**(function (i) {**

**setTimeout(function () {**

**console.log(i + ' ');**

**}, 100);**

**}(i));**

**}**

现有一个**Page**类**,**其原型对象上有许多以**post**开头的方法**(**如**postMsg);**另有一拦截函数**chekc,**只返回**ture**或**false.**请设计一个函数**,**该函数应批量改造原**Page**的**postXXX**方法**,**在保留其原有功能的同时**,**为每个**postXXX**方法增加拦截验证功能**,**当**chekc**返回**true**时继续执行原**postXXX**方法**,**返回**false**时不再执行原**postXXX**方法

**function Page() {}**

**Page.prototype = {**

**constructor: Page,**

**postA: function (a) {**

**console.log('a:' + a);**

**},**

**postB: function (b) {**

**console.log('b:' + b);**

**},**

**postC: function (c) {**

**console.log('c:' + c);**

**},**

**check: function () {**

**return Math.random() > 0.5;**

**}**

**}**

**function checkfy(obj) {**

**for (var key in obj) {**

**if (key.indexOf('post') === 0 && typeof obj[key] === 'function') {**

**(function (key) {**

**var fn = obj[key];**

**obj[key] = function () {**

**if (obj.check()) {**

**fn.apply(obj, arguments);**

**}**

**};**

**}(key));**

**}**

**}**

**} // end checkfy()**

**checkfy(Page.prototype);**

**var obj = new Page();**

**obj.postA('checkfy');**

**obj.postB('checkfy');**

**obj.postC('checkfy');**