**知识点**

数据类型 、 运算 、 对象 、 function 、 继承 、 闭包 、 作用域 、 原型链 、 事件 、 RegExp 、 JSON 、 Ajax 、 DOM 、 BOM 、 内存泄漏 、 跨域、 异步加载 、 模板引擎 、 前端MVC 、 前端MVVM 、 路由 、 模块化 、Canvas 、 jQuery 、 ECMAScript 2015（ES6） 、 Node.js 、AngularJS 、 React 、 CommonJS 、 AMD 、 CMD ......

* 介绍一下 JS 的基本数据类型。

Undefined、Null、Boolean、Number、String

* 介绍一下 JS 有哪些内置对象。
* Object 是 JavaScript 中所有对象的父对象
* 数据封装类对象：Object、**Array**、Boolean、Number、String

其他对象：Function、Argument、Math、Date、RegExp、Error

* 列举几条 JavaScript 的基本代码规范。
* （1）不要在同一行声明多个变量
* （2）如果你不知道数组的长度，使用 **push**
* （3）请使用 ===/!== 来比较 true/false 或者数值
* （4）对字符串使用单引号 ''(因为大多时候我们的字符串。特别html会出现")
* （5）使用对象字面量替代 new Array 这种形式
* （6）绝对不要在一个非函数块里声明一个函数，把那个函数赋给一个变量。浏览器允许你这么做，但是它们解析不同
* （7）不要使用全局函数
* （8）总是使用 var 来声明变量，如果不这么做将导致产生全局变量，我们要避免污染全局命名空间
* （9）Switch 语句必须带有 default 分支
* （10）使用 /\*\*...\*/ 进行多行注释，包括描述，指定类型以及参数值和返回值
* （11）函数不应该有时候有返回值，有时候没有返回值
* （12）语句结束一定要加分号
* （13）for 循环必须使用大括号
* （14）if 语句必须使用大括号
* （15）for-in 循环中的变量应该使用 var 关键字明确限定作用域，从而避免作用域污染
* （16）避免单个字符名，让你的变量名有描述意义
* （17）当命名对象、函数和实例时使用驼峰命名规则
* （18）给对象原型分配方法，而不是用一个新的对象覆盖原型，覆盖原型会使继承出现问题

（19）当给事件附加数据时，传入一个哈希而不是原始值，这可以让后面的贡献者加入更多数据到事件数据里，而不用找出并更新那个事件的事件处理器

* 介绍一下 JavaScript 原型，原型链，它们有何特点？
* 每个对象都会在其内部初始化一个属性，就是prototype(原型)，当我们访问一个对象的属性时，如果这个对象内部不存在这个属性，那么他就会去prototype里找这个属性，这个prototype又会有自己的prototype，
* 于是就这样一直找下去，也就是我们平时所说的原型链的概念。
* 关系：instance.constructor.prototype = instance.\_\_proto\_\_
* *//*
* 特点：JavaScript对象是通过引用来传递的，我们创建的每个新对象实体中并没有一份属于自己的原型副本，当我们修改原型时，与之相关的对象也会继承这一改变。
* *//*
* 当我们需要一个属性时，JavaScript引擎会先看当前对象中是否有这个属性，如果没有的话，就会查找它的prototype对象是否有这个属性，如此递推下去，一致检索到Object内建对象。
* **function** **Func**(){}
* Func.prototype.name = "Xiaosong";
* Func.prototype.getInfo = **function**() {
* **return** **this**.name;
* }
* **var** person = **new** Func();
* console.log(person.getInfo());
* *//"Xiaosong"*
* console.log(Func.prototype);

*//Func { name = "Xiaosong", getInfo = function() }*

* JavaScript 有几种类型的值？能否画一下它们的内存图？
* 栈：原始数据类型（Undefined，**Null**，Boolean，Number，String）
* 堆：引用数据类型（对象、数组、函数）
* 两种类型的区别：存储位置不同
* *//*
* 原始数据类型直接存储在栈(stack)中的简单数据段，占据空间小、大小固定，属于被频繁使用数据，所以放入栈中存储；

引用数据类型存储在堆(heap)中的对象,占据空间大、大小不固定,如果存储在栈中，将会影响程序运行的性能；引用数据类型在栈中存储了指针，该指针指向堆中该实体的起始地址。当解释器寻找引用值时，会首先检索其在栈中的地址，取得地址后从堆中获得实体。

* JavaScript 如何实现继承？
* (1)构造继承
* (2)原型继承
* (3)实例继承
* (4)拷贝继承
* *//*
* 原型prototype机制或apply和call方法去实现较简单，建议使用构造函数与原型混合方式。
* **function** **Parent**() {
* **this**.name = 'song';
* }
* **function** **Child**() {
* **this**.age = 28;
* }
* Child.prototype = **new** **Parent**(); *//通过原型,继承了Parent*
* *//*
* **var** demo = **new** Child()l;
* alert(demo.age);

alert(demo.name); *//得到被继承的属性*

* JavaScript 有哪几种创建对象的方式？
* javascript创建对象简单的说,无非就是使用内置对象或各种自定义对象，当然还可以用JSON；但写法有很多种，也能混合使用。
* *//*
* (1)对象字面量的方式
* person={firstname:"Mark",lastname:"Yun",age:25,eyecolor:"black"};
* (2)用**function**来模拟无参的构造函数
* **function** **Person**(){}
* **var** person = **new** Person(); *//定义一个function，如果使用new"实例化",该function可以看作是一个Class*
* person.name = "Xiaosong";
* person.age = "23";
* person.work = **function**() {
* alert("Hello " + person.name);
* }
* person.work();
* (3)用**function**来模拟参构造函数来实现（用**this**关键字定义构造的上下文属性）
* **function** **Person**(name,age,hobby) {
* **this**.name = name; *//this作用域：当前对象*
* **this**.age = age;
* **this**.work = work;
* **this**.info = **function**() {
* alert("我叫" + **this**.name + "，今年" + **this**.age + "岁，是个" + **this**.work);
* }
* }
* **var** Xiaosong = **new** Person("WooKong",23,"程序猿"); *//实例化、创建对象*
* Xiaosong.info(); *//调用info()方法*
* (4)用工厂方式来创建（内置对象）
* **var** jsCreater = **new** Object();
* jsCreater.name = "Brendan Eich"; *//JavaScript的发明者*
* jsCreater.work = "JavaScript";
* jsCreater.info = **function**() {
* alert("我是"+**this**.work+"的发明者"+**this**.name);
* }
* jsCreater.info();
* (5)用原型方式来创建
* **function** **Standard**(){}
* Standard.prototype.name = "ECMAScript";
* Standard.prototype.event = **function**() {
* alert(**this**.name+"是脚本语言标准规范");
* }
* **var** jiaoben = **new** Standard();
* jiaoben.event();
* (6)用混合方式来创建
* **function** **iPhone**(name,event) {
* **this**.name = name;
* **this**.event = event;
* }
* iPhone.prototype.sell = **function**() {
* alert("我是"+**this**.name+"，我是iPhone5s的"+**this**.event+"~ haha!");
* }
* **var** SE = **new** iPhone("iPhone SE","官方翻新机");

SE.sell();

* eval 是做什么的？
* 它的功能是把对应的字符串解析成JS代码并运行；

应该避免使用**eval**，因为不安全，非常耗性能（2次，一次解析成js语句，一次执行）。

* 什么是 document 对象？什么是 window 对象？
* null 和 undefined 有何区别？
* null 表示一个对象被定义了，值为“空值”；
* undefined 表示不存在这个值。
* *//*
* **typeof** undefined
* *//"undefined"*
* undefined :是一个表示"无"的原始值或者说表示"缺少值"，就是此处应该有一个值，但是还没有定义。当尝试读取时会返回 undefined；
* 例如变量被声明了，但没有赋值时，就等于undefined。
* *//*
* **typeof** null
* *//"object"*
* null : 是一个对象(空对象, 没有任何属性和方法)；
* 例如作为函数的参数，表示该函数的参数不是对象；
* *//*
* 注意：

在验证null时，一定要使用　=== ，因为 == 无法分别 null 和　undefined

* 能否写一个通用的事件侦听器函数？
* *//Event工具集，from:github.com/markyun*
* markyun.Event = {
* *//页面加载完成后*
* readyEvent: **function**(fn) {
* **if** (fn == null) {
* fn = document;
* }
* **var** oldonload = window.onload;
* **if** (**typeof** window.onload != 'function') {
* window.onload = fn;
* }**else**{
* window.onload = **function**() {
* oldonload();
* fn();
* };
* }
* },
* *//视能力分别使用 demo0 || demo1 || IE 方式来绑定事件*
* *//参数：操作的元素，事件名称，事件处理程序*
* addEvent: **function**(element,type,handler) {
* **if** (element.addEventListener) {
* *//事件类型、需要执行的函数、是否捕捉*
* element.addEventListener(type,handler,false);
* }**else** **if** (element.attachEvent) {
* element.attachEvent('on' + type, **function**() {
* handler.call(element);
* });
* }**else** {
* element['on' + type] = handler;
* }
* },
* *//移除事件*
* removeEvent: **function**(element,type,handler) {
* **if** (element.removeEventListener) {
* element.removeEventListener(type,handler,false);
* }**else** **if** (element.datachEvent) {
* element.datachEvent('on' + type,handler);
* }**else**{
* element['on' + type] = null;
* }
* },
* *//阻止事件（主要是事件冒泡，因为IE不支持事件捕获）*
* stopPropagation: **function**(ev) {
* **if** (ev.stopPropagation) {
* ev.stopPropagation();
* }**else** {
* ev.cancelBubble = true;
* }
* },
* *//取消事件的默认行为*
* preventDefault: **function**(event) {
* **if** (event.preventDefault) {
* event.preventDefault();
* }**else**{
* event.returnValue = false;
* }
* },
* *//获取事件目标*
* getTarget: **function**(event) {
* **return** event.target || event.srcElemnt;
* },
* *//获取event对象的引用，取到事件的所有信息，确保随时能使用event；*
* getEvent: **function**(e) {
* **var** ev = e || window.event;
* **if** (!ev) {
* **var** c = **this**.getEvent.caller;
* **while**(c) {
* ev = c.argument[0];
* **if** (ev && Event == ev.constructor) {
* **break**;
* }
* c = c.caller;
* }
* }
* retrun ev;
* }

};

* ["1","2","3"].map(parseInt) 的答案是多少？
* [1,NaN,NaN]
* 因为 parseInt 需要两个参数(val,radix)，其中 radix 表示解析时用的基数。

map 传了3个(element,index,array)，对应的 radix 不合法导致解析失败。

* 事件是什么？IE与火狐的事件机制有何区别？如何阻止冒泡？
* (1)我们在网页中的某个操作（有的操作对应多个事件）。例如：当我们点击一个按钮就会产生一个事件。是可以被 JavaScript 侦测到的行为。
* (2)事件处理机制：IE是事件冒泡、Firefox同时支持两种事件模型，也就是：捕获型事件和冒泡型事件；

(3)ev.stopPropagation();（旧ie的方法 ev.cancelBubble = true;）

* 什么是闭包(closure)，为什么要用它？
* 闭包是指有权访问另一个函数作用域中变量的函数，创建闭包的最常见的方式就是在一个函数内创建另一个函数，通过另一个函数访问这个函数的局部变量，利用闭包可以突破作用链域，将函数内部的变量和方法传递到外部。
* *//*
* 闭包特性：
* (1)函数内再嵌套函数
* (2)内部函数可以引用外层的参数和变量
* (3)参数和变量不会被垃圾回收机制回收
* *//li节点的onclick事件都能正确的弹出当前被点击的li索引*
* <ul>
* <li> index = 0 </li>
* <li> index = 1 </li>
* <li> index = 2 </li>
* <li> index = 3 </li>
* </ul>
* <script type="text/javascript">
* **var** nodes = document.getElementsByTagName('li');
* **for**(i = 0;i<nodes.length;i+=1) {
* nodes[i].onclick = **function**() {
* console.log(i+1); *//不使用闭包的话，值每次都是4*
* }(4);
* }

</script>

* JavaScript 代码中的 "use strict"; 是什么意思？使用它的区别是什么？
* **use** strict是一种ECMAscript 5 添加的（严格）运行模式,这种模式使得 Javascript 在更严格的条件下运行,使JS编码更加规范化的模式,消除Javascript语法的一些不合理、不严谨之处，减少一些怪异行为。
* 默认支持的糟糕特性都会被禁用，比如不能用with，也不能在意外的情况下给全局变量赋值;
* 全局变量的显示声明,函数必须声明在顶层，不允许在非函数代码块内声明函数,arguments.callee也不允许使用；
* 消除代码运行的一些不安全之处，保证代码运行的安全,限制函数中的arguments修改，严格模式下的**eval**函数的行为和非严格模式的也不相同;
* 提高编译器效率，增加运行速度；

为未来新版本的Javascript标准化做铺垫。

* new 操作符具体干了什么呢？
* (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)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** 。
* *//*
* **var** obj = {};
* obj.\_\_proto\_\_ = Base.prototype;

Base.call(obj);

* 用原生的 JavaScript 实现过什么功能吗？
* JavaScript 中，有一个函数，执行对象查找时，永远不会去查找原型，这个函数是哪个？
* hasOwnProperty
* *//*
* JavaScript 中 hasOwnProperty 函数方法是返回一个布尔值，指出一个对象是否具有指定名称的属性。此方法无法检查该对象的原型链中是否具有该属性；该属性必须是对象本身的一个成员。
* *//*
* 使用方法：
* **object**.hasOwnProperty(proName)
* 其中参数**object**是必选项，一个对象的实例。
* proName是必选项，一个属性名称的字符串值。
* *//*

如果 **object** 具有指定名称的属性，那么JavaScript中hasOwnProperty函数方法返回 **true**，反之则返回 **false**。

* 你对 JSON 了解吗？
* JSON(JavaScript Object Notation)是一种轻量级的数据交换格式。
* 它是基于JavaScript的一个子集。数据格式简单，易于读写，占用带宽小。
* 如：

{"age":"12", "name":"back"}

* js延迟加载的方式有哪些？

defer和async、动态创建DOM方式（用得最多）、按需异步载入js

* Ajax 是什么？如何创建一个 Ajax ？
* ajax的全称：Asynchronous Javascript **And** XML。
* 异步传输+js+xml。
* 所谓异步，在这里简单地解释就是：向服务器发送请求的时候，我们不必等待结果，而是可以同时做其他的事情，等到有了结果它自己会根据设定进行后续操作，与此同时，页面是不会发生整页刷新的，提高了用户体验。
* *//*
* (1)创建XMLHttpRequest对象,也就是创建一个异步调用对象
* (2)创建一个新的HTTP请求,并指定该HTTP请求的方法、URL及验证信息
* (3)设置响应HTTP请求状态变化的函数
* (4)发送HTTP请求
* (5)获取异步调用返回的数据

(6)使用JavaScript和DOM实现局部刷新

* 同步和异步的区别？
* 同步的概念应该是来自于操作系统中关于同步的概念:
* 不同进程为协同完成某项工作而在先后次序上调整(通过阻塞,唤醒等方式)。同步强调的是顺序性，谁先谁后；异步则不存在这种顺序性。
* *//*
* 同步：浏览器访问服务器请求，用户看得到页面刷新，重新发请求,等请求完，页面刷新，新内容出现，用户看到新内容,进行下一步操作。
* *//*

异步：浏览器访问服务器请求，用户正常操作，浏览器后端进行请求。等请求完，页面不刷新，新内容也会出现，用户看到新内容。

* 如何解决跨域问题？

jsonp、iframe、window.name、window.postMessage、服务器上设置代理页面

* 页面编码和被请求的资源编码如果不一致如何处理？
* 谈一谈你对 ECMAScript6 的了解？
* ECMAScript 6 是JavaScript语言的下一代标准，已经在2015年6月正式发布了。它的目标，是使得JavaScript语言可以用来编写复杂的大型应用程序，成为企业级开发语言。

标准的制定者有计划，以后每年发布一次标准，使用年份作为标准的版本。因为当前版本的ES6是在2015年发布的，所以又称ECMAScript 2015。也就是说，ES6就是ES2015

* ECMAScript 6 怎么写 class ，为何会出现 class？
* ES6的**class**可以看作只是一个语法糖，它的绝大部分功能，ES5都可以做到，新的**class**写法只是让对象原型的写法更加清晰、更像面向对象编程的语法而已。
* *//定义类*
* **class** Point {
* constructor(x,y) { *//构造方法*
* **this**.x = x; *//this关键字代表实例对象*
* **this**.y = y;
* }
* toString() {
* **return** '(' + **this**.x + ',' + **this**.y + ')';
* }

}

* 异步加载 JS 的方式有哪些？
* (1)defer，只支持 IE
* (2)async:

(3)创建 script，插入到 DOM 中，加载完毕后 callBack

* document.write 和 innerHTML 有何区别？
* document.**write** 只能重绘整个页面

innerHTML 可以重绘页面的一部分

* DOM 操作——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
* appendChild()
* removeChild()
* replaceChild()
* insertBefore() *//在已有的子节点前插入一个新的子节点*
* (3)查找
* getElementsByTagName() *//通过标签名称*
* getElementsByName() *//通过元素的Name属性的值(IE容错能力较强，会得到一个数组，其中包括id等于name值的)*

getElementById() *//通过元素Id，唯一性*

* 数组和对象有哪些原生方法？能否列举一下？
* 如何编写高性能 JavaScript ？

详细文章： ***[浅谈编写高性能的Javascript代码](http://developer.51cto.com/art/200906/131335.htm" \t "_blank)***

* 哪些操作会造成内存泄漏？
* 内存泄漏是指任何对象在您不再拥有或需要它之后任然存在。
* 垃圾回收器定期扫描对象，并计算引用了每个对象的其他对象的数量，如果一个对象的引用数量为0（没有其他对象引用过该对象），或对该对象的惟一引用是循环的，那么该对象的内存即可回收。
* *//*
* setTimeout 的第一个参数使用字符串而非函数的话，会引发内存泄漏。

闭包、控制台日志、循环（在两个对象彼此引用且彼此保留时，就会产生一个循环）

* 是否看过 jQuery 的源码？能否简单概括一下它的实现原理？
* jQuery.fn 的 init 方法返回的 this 指的是什么对象？为什么要返回 this ？
* jQuery 中如何将数组转化为 json 字符串，然后再转化回来？
* jQuery 中没有提供这个功能，所以需要先编写两个 jQuery 的扩展：
* $.fn.stringifyArray = function(**array**) {
* **return** JSON.stringify(**array**)
* }
* $.fn.parseArray = function(**array**) {
* **return** JSON.parse(**array**)
* }
* *//*
* 然后调用:

$("").stringifyArray(**array**)

* jQuery 的属性拷贝（extend）的实现原理是什么？如何实现深拷贝？
* jQuery.extend 与 jQuery.fn.extend 有何区别？
* jQuery 的队列是如何实现的？队列可以用在哪些地方？
* jQuery 中一个对象可以同时绑定多个事件，这是如何实现的？
* 是否了解针对 jQuery 性能的优化方法？
* 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++) {}

* jQuery 与 jQuery UI 有何区别？
* `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。

`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

* jQuery UI 如何自定义组件？
* 如何判断当前脚本运行在浏览器还是 node 环境中？（阿里）

通过判断 **Global** 对象是否为 window ，如果不为 window ，当前脚本没有运行在浏览器中

* 什么是“前端路由”？什么时候适合使用“前端路由”？“前端路由”有哪些优点和缺点？
* 怎样用js实现千位分隔符？
* 正则 + replace
* **function** **commafy**(num) {
* num = num + '';
* **var** reg = /(-?d+)(d{3})/;
* **if** (reg.test(num)) {
* num = num.replace(reg, '$1,$2');
* }
* **return** num;

}

* 检测浏览器版本有哪些方式？
* 功能检测、userAgent 特征检测
* 比如：navigator.userAgent

*//"Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_10\_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/41.0.2272.101 Safari/537.36"*

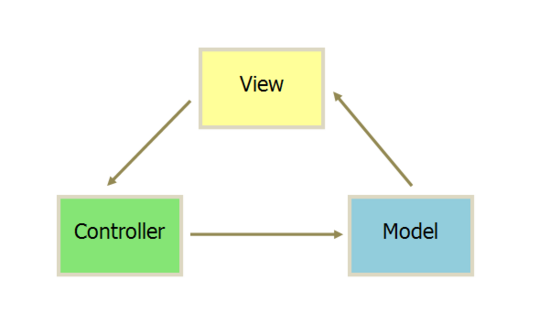
* 谈谈你对 JavaScript 中的模块规范 CommonJS、AMD、CMD 的了解？
* *//个人拙见*
* | CommonJS | AMD | CMD |
* |--------------|---------|---------|

| Node.js |RequireJS| SeaJS |

详细文章： ***[浅析JS中的模块规范（CommonJS，AMD，CMD）](http://www.2cto.com/kf/201411/348276.html" \t "_blank)*** 、 ***[关于 CommonJS AMD CMD UMD](http://my.oschina.net/felumanman/blog/263330?p=1" \t "_blank)***

* 前端 MVC、MVVM

1、MVC



MVC

模型（Model）：数据保存

视图（View）：用户界面

控制器（Controller）：业务逻辑

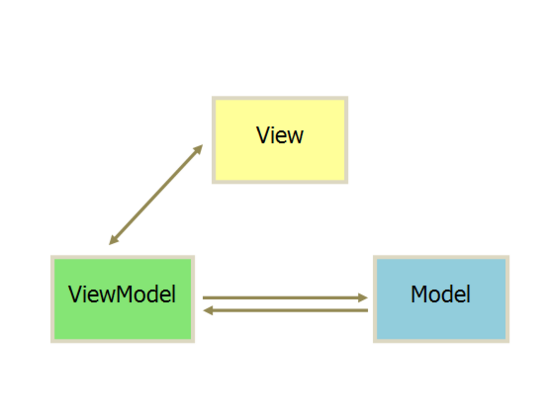
(1)View 传送指令到 Controller

(2)Controller 完成业务逻辑后，要求 Model 改变状态

(3)Model 将新的数据发送到 View ，用户得到反馈

所有通信都是单向的。

2、MVVM



MVVM

模型（Model）

视图（View）

视图模型（ViewModel）

(1)各部分间都是双向通信

(2)View 与 Model 不发生联系，都通过 ViewModel 传递

(3)View 非常薄，不部署任何业务逻辑，称为“被动视图”（Passive View），即没有任何主动性；而 ViewModel 非常厚，所有逻辑都部署在那里

采用双向绑定（data-binding）：View 的变动，自动反映在 ViewModel ，反之亦然。

## 1. 上下左右居中

<style type="text/css">

**#test**{

position:absolute;

top:50%;

left: 50%;

transform: translate(-50%,-50%);

}

</style>

<div id="test" style="border-color: aqua;width: 100px;height: 100px;border: 2px solid;"></div>

## 2. 两栏，左边栏宽度固定100px，右边栏宽度自适应

思路：左边栏宽度固定，设置左浮动，右边栏设置margin-left属性，值为左边栏的宽度。

<style type="text/css">

\*{

margin: 0;

padding: 0;

}

**#left**{

float:left;

width:100px;

height: 100px;

border: solid #999 1px;

}

**#right**{

height: 100px;

margin-left: 100px;

border: solid #777 1px;

}

</style>

<div id="left"></div>

<div id="right"></div>

## 3. 盒模型

盒模型可以帮助块级元素的定位和计算面积，简单来说W3C的盒模型包括content、padding、border。而IE得盒模型则有些不同，表现在设置宽高的时候，W3C模型下只包含content区域，而IE包括所有区域。这样，在IE下，设置了宽高之后可以很放心的设定padding，但是对于W3C模型下，设置了宽高之后还要计算padding和border的区域。因此有一个方法叫做box-sizing:border-box。border-box将border和padding区域一同算进设定的宽高内。这样的话添加如下代码，就可以设定padding而用不担心超出设置区域。

div{

box-sizing:border-box;

-moz-box-sizing:border-box; */\* Firefox \*/*

-webkit-box-sizing:border-box; */\* Safari \*/*

}

## 4. 几种方法生成三个并排的大小相等的元素

一个方法为tabel-cell，这种方法就是将每一个元素设置的尽可能的大，但由于自适应的特性，这三个元素还是会乖乖地等分的排在一行里。

<style type="text/css">

\*{

margin: 0;

padding: 0;

}

**.left**,**.right**,**.middle**{

display:table-cell;

width: 10000px;

border: 1px solid red;

height:100px;

}

</style>

方法二：用百分比控制

\*{

margin: 0;

padding: 0;

}

**.left**,**.right**,**.middle**{

float:left;

width:33.3%;

margin-right: -4px//可以按照实际调整。

}

## 5. CSS3的一些新特性

1. border-radius让圆角成为方便的存在   
   怎样将方的通过圆角变成圆的？   
   border-radius:50%;
2. box-shadow(渐变)也是咱常用的属性之一。设置水平、垂直的位移，阴影的模糊、尺寸还有颜色，就可以为元素添加阴影。   
   box-shadow: 2px 2px 3px #777;
3. 通过设置图片路径，图片的切割方式、还有边框的宽度、溢出，以及平铺或拉伸等呈现方式，可以做出一个漂亮精致的边框。   
   border-image: url(border-image.png) 30/10px/10px round;
4. 动画：transition、transform和animation的区别?   
   transform是指转换，可以将元素移动、旋转、倾斜、拉伸。   
   translate()，从当前位置移动到由给定left和top值的位置。这个例子中，div向右下移动，如果想要往左上移动，则要设置为负值。
5. div{
6. transform: translate(50px,100px);
7. -ms-transform: translate(50px,100px); */\* IE 9 \*/*
8. -webkit-transform: translate(50px,100px); */\* Safari and Chrome \*/*
9. -o-transform: translate(50px,100px); */\* Opera \*/*
10. -moz-transform: translate(50px,100px); */\* Firefox \*/*

}

rotate()，将元素旋转到给定角度，单位为deg（degree角度），在这个例子中为顺时针30度。负值的话为逆时针旋转。

div{

transform: rotate(30deg);

-ms-transform: rotate(30deg); */\* IE 9 \*/*

-webkit-transform: rotate(30deg); */\* Safari and Chrome \*/*

-o-transform: rotate(30deg); */\* Opera \*/*

-moz-transform: rotate(30deg); */\* Firefox \*/*

}

skew() ，将元素倾斜到给定角度，单位也是deg，分别围绕着X轴和Y轴翻转。这个例子中，在X轴顺时针翻转30度，Y轴顺时针翻转20度，负值为逆时针翻转。

div{

transform: skew(30deg,20deg);

-ms-transform: skew(30deg,20deg); */\* IE 9 \*/*

-webkit-transform: skew(30deg,20deg); */\* Safari and Chrome \*/*

-o-transform: skew(30deg,20deg); */\* Opera \*/*

-moz-transform: skew(30deg,20deg); */\* Firefox \*/*

}

scale()，将元素拉伸到指定的倍数，同样设定了X与Y两个方向。这个例子中把宽度拉伸2倍，高度拉伸4倍。有意思的是，负值不是缩小，而是翻转，既水平翻转和垂直翻转。

div{

transform: scale(2,4);

-ms-transform: scale(2,4); */\* IE 9 \*/*

-webkit-transform: scale(2,4); */\* Safari 和 Chrome \*/*

-o-transform: scale(2,4); */\* Opera \*/*

-moz-transform: scale(2,4); */\* Firefox \*/*

}

transform并没有变化的过程，而是直接生成最终效果。transform还有3D方法，就是多了一个Z轴

transition是指过渡，可以动画般显示出一个从样式到样式之间的过渡。transform的属性包括一个你设定过渡的CSS属性，持续时间，时间曲线还有过渡开始的时间。

div{

transition-property: width;

transition-duration: 1s;

transition-timing-function: linear;

transition-delay: 2s;

*/\* Firefox 4 \*/*

-moz-transition-property:width;

-moz-transition-duration:1s;

-moz-transition-timing-function:linear;

-moz-transition-delay:2s;

*/\* Safari 和 Chrome \*/*

-webkit-transition-property:width;

-webkit-transition-duration:1s;

-webkit-transition-timing-function:linear;

-webkit-transition-delay:2s;

*/\* Opera \*/*

-o-transition-property:width;

-o-transition-duration:1s;

-o-transition-timing-function:linear;

-o-transition-delay:2s;

}

animation专门做动画的animation就复杂多了：@keyframes可以有from to，也可以是百分比表示时间帧。

## 6. 获取的元素，然后循环为每一个添加mouseover和mouseout事件。

**var** elementList=document.querySelectorAll("a");

**var** listLen=elementList.length;

**function** **handler** (element,color){

**return** **function**(){

element.style.color=color;

};

}

*//添加事件方法*

**function** **addEvent**(element,type,handler){

**if**(element.addEventListener){

element.addEventListener(type,handler,false);

}

**else** **if**(element.attachEvent){

element.attachEvent("on"+type,handler);

}

}

*//循环为元素添加事件*

**for**(i=0;i<listLen;i++){

addEvent(elementList[i],"mouseover",handler(elementList[i],"red"));

addEvent(elementList[i],"mouseout",handler(elementList[i],"black"));

}

## 7. 事件委托

什么是事件委托：通俗的讲，事件就是onclick，onmouseover，onmouseout，等就是事件，委托呢，就是让别人来做，这个事件本来是加在某些元素上的，然而你却加到别人身上来做，完成这个事件。

也就是：利用冒泡的原理，把事件加到父级上，触发执行效果。

好处呢：1，提高性能。

这样我们就可以做到li上面添加鼠标事件。

但是如果说我们可能有很多个li用for循环的话就比较影响性能。

## 8. 添加一个类似email中checklist的双向联动

做一个类似email里选择发件人的东西，勾选的时候添加到发件人，取消勾选的时候从发件人里面删除。由于是双向联动，因此要在发件人里点击的时候可以删除，并且取消勾选。好像说的有点乱，看下面代码。其实主要还是事件绑定，然后添加节点，改变属性。

<div class="container">

<div class="showbox"></div>

<input type="checkbox" name="email" value="i@myj.name">Kathy</input>

<input type="checkbox" name="email" value="i@zjy.name">Visper</input>

</div>

var emails=document.querySelectorAll("[name=email]");

var showBox=document.querySelector(".showbox");

var length=emails.length;

//添加事件

function addEvent(element,type,handler){

if(element.addEventListener){

element.addEventListener(type,handler,false);

}

else if(element.attachEvent){

element.attachEvent("on"+type,handler);

}

}

//循环，为每个checkbox添加事件

for(var i=0;i<length;i++){

addEvent(emails[i],"click",function(){

//判断是否被选中

if(this.checked){

var Emailvalue = this.value;

var newBtn=document.createElement("input");

newBtn.setAttribute("type","button");

newBtn.value=Emailvalue;

showBox.appendChild(newBtn);

}

//若取消勾选，则删除相应的button

else{

var Emailvalue = this.value;

var emailBtn=document.querySelector("[type=button][value='"+Emailvalue+"']");

showBox.removeChild(emailBtn);

}

//每次重新调用以更新button

startBtn();

})

}

//获取button列表，为button添加事件

function startBtn(){

var emailBtn=document.querySelectorAll("[type=button]");

if(emailBtn.length){

for(var i=0;i<emailBtn.length;i++){

addEvent(emailBtn[i],"click",function(){

var btnvalue=this.value;

var cancelChoose=document.querySelectorAll("[value='"+btnvalue+"']");

//删除button并取消勾选相应checkbox

for(var j=0;j<cancelChoose.length;j++){

if(cancelChoose[j].getAttribute("type")=="button"){

showBox.removeChild(cancelChoose[j]);

}

else if(cancelChoose[j].getAttribute("type")=="checkbox"){

cancelChoose[j].checked=false;

}

}

})

}

}

}

## 9. Array的方法

isArray()，判断是否为Array类型。join()，将Array类型转换为字符串类型,并用join内的符号间隔开。

队列，栈的用法

push()，将数据项放到数组的后面。

pop()，将数组的最后一项删除并返回。

shift()，将数组的第一项删除并返回。

unshift()，将数据项放到数组的前面。

重排序的方法

reverse()，将数组逆序排列。

sort()，将数组由小到大排列。默认是按照字符串比较，也就是说15比5小。要按照数字排列的话，要设定方法：

**var** numlist = [0, 5, 15, 20, 10];

**function** **compare** (value1, value2){

value2-value1;

}

**var** listsort = numlist.sort(compare);

alert( listsort );*//0, 5, 10, 15, 20 按照数值从小到大*

操作Array的方法

concat()，在原有meat数组上创建新数组

slice()，截取从指定位置开始到结束为止(不包含结束位置)的数组项赋给新数组，如果没有第二个参数，则截取从指定位置开始到结束的数组项赋给新数组

splice()，从指定位置(第一个参数)删除指定个数(第二个参数)并插入指定项目(’第三个参数”)

位置方法

indexOf()，从前向后查找位置索引。

lastIndexOf()，从后向前查找位置索引。

这两个方法支持第二个参数，既指定位置然后按照方法向前或向后查找位置。

遍历方法

every()中，每一项运行给定函数，每一项都返回true,??则返回true。

filter()中，每一项运行给定函数，返回true的项。

forEach()中，每一项运行给定函数，无返回。

map()中，每一项运行给定函数，返回调用结果组成的数组。

some()中，每一项运行给定函数，任意一项返回true，?则返回true。

归并方法

reduce()和reduceRight()都接受四个参数：前一个值，当前值，索引和数组对象。

reduce()是从左向右归并，当第一个值和第二个值运行过之后，结果将作为下一次运行的前一个值。

reduceRight()则是从右向左，归并方式与reduce()一样。

## 10. 给定字符串“abcba”，处理得到第一个不重复字母。

function firstOnlyChar(str) {

**var** str = str || "",

i=0,

k='',

\_**char** = "",

charMap = {},

result = {name:"",index:str.length};

**for**(i=0;i<str.length;i++){

\_**char** = str.charAt(i);

**if**(charMap[\_**char**]!=undefined){

charMap[\_**char**]=-1;

}**else**{

charMap[\_**char**]=i;

}

}

**for**(k **in** charMap){

**if**(charMap[k]<0){

**continue**;

}

**if**(result.index>charMap[k]){

result.index=charMap[k];

result.name=k;

}

}

**return** result.name;

}

## 11. AJAX过程，AJAX中如何区分get和post，get和post的区别

AJAX算是听人说的最多的一个技术，无需加载整个页面的情况下进行局部更新。再用AJAX技术与服务器交换数据时，首先创建一个XMLHttpRequest对象，在特立独行的IE里是ActiveXObject。

接着我们用open()方法和send()方法，使用get和post会有些区别。然后，请求会发送给服务器，服务器会响应触发onreadystatechange事件，当readyState=4（请求完成）且status=200时，说明服务器已经就绪。然后用responseText或responseXML属性去获得字符串或XML格式的数据。

**var** xmlhttp;

**if** (window.XMLHttpRequest){

xmlhttp=**new** XMLHttpRequest();

}

**else**{

*//奇葩的IE5、6*

xmlhttp=**new** ActiveXObject("Microsoft.XMLHTTP");

}

xmlhttp.open("GET","response.php",true);

xmlhttp.send();

xmlhttp.onreadystatechange=**function**(){

**if** (xmlhttp.readyState==4 && xmlhttp.status==200){

document.getElementById("showBox").innerHTML=xmlhttp.responseText;

}

}

在AJAX中区别get和post，就要详谈open()方法和send()方法。

get请求常用于向服务器查询信息。

在get请求中，open()的第一个参数为“get”，第二个参数为URL，第三个参数为是否为异步。由于get请求的URL中通常带有参数，这些参数必须要使用encodeURIComponent()方法进行编码，否则会有格式错误。这样send()方法内部的参数可以传入null，传入的参数也会转到URL中。

post请求常用于向服务器提交信息。

在post请求中，open()的第一个参数为“post”，URL中不含有参数，第三个参数一样。post请求的send()方法中需要传入提交数据，可以使用XML或者字符串。在提交表单时，首先要设定请求头部的Content-Type为 application/x-www-form-urlencoded。

xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");然后将传入的表单信息序列化，方便后端的程序获取数据。对于表单数据序列化，FormData对象可以直接放入send()内:

<form id=“user-info”>

用户名：<input type="text" name="username"/>

年龄：<input type="text" name="年龄"/>

</form>

var form = document.getElementById("user-info");

xhr.send(new FormData(form));

前面提过，get方式中url包含参数，而post中url是“干干净净”的。因为用处不同，post在提交内容的时候没有大小限制。而且post在发送时有内容体，那些提交的信息放在内容体内。

## 12. JSONP 跨域的特性

由于跨域安全策略，XMLHttpRequest不能跨域请求，这是AJAX的一个主要限制。JSONP是通过<\script>元素。因为我们知道，<\script>可以跨域引用js文件。返回的数据格式为JSON，创建一个回调函数来相应返回的数据。

var script = document.createElement("script");

 script.src = "http://pvoutput.org/service2v?Apikey=123&SystemID=123&callback=**handler**”;

document.body.appendChild(script);

//回调函数

function **handler**(response){

var responseContent={

diatotime:response.datetime,

usedEnergy:response.used.Energy,

…

}

return responseContent；

}

除了JSONP，用img标签也可以跨域通信。

**var** img = **new** Image();

img.src = "http://www.example.com/id=123”;

CORS是跨域请求的另一个方法。在IE浏览器中为XDomainRequest对象，而在其他浏览器中则继续用XMLHttpRequest。区别在于，XDomainRequest对象的open()只有两个参数：请求类型和URL，因为都是异步执行。而其他浏览器在使用open()时，URL要使用绝对路径，既可以跨域请求。

## 13.提高网页性能的方法

将小图片放在一起减少HTTP请求。然后想到将script和css从外部引入，然后script放在文档底部。

除此之外，总结一下：在内容上压缩，在请求上减少。压缩图片、脚本和CSS，使用CSS精灵，优化图片，尽量使用<link>引用CSS。在请求上避免重定向，尽量使用AJAX尽量使用GET请求。

## 14.Nodejs性能优化

### 1. 并行

创建 Web 应用的时候，你可能要多次调用内部 API 来获取各种数据。比如说，假设在 Dashboard 页面上，你要执行下面这几个调用:

用户信息 -getUserProfile().

当前活动 -getRecentActivity().

订阅内容 -getSubscriptions().

通知内容 -getNotifications().

为了拿到这些信息，你应该会为每个方法创建独立的中间件，然后将它们链接到 Dashboard 路由上。不过问题是，这些方法的执行是线性的，上一个没结束之前下一个不会开始。可行解决案是并行调用它们。

如你所知由于异步性，Node.js 非常擅长并行调用多个方法。我们不能暴殄天物。我上面提到的那些方法没有依赖性，所以我们可以并行执行它们。这样我们可以削减中间件数量，大幅提高速度。

我们可以用 async.js 来处理并行，它是一个专门用来调教 JavaScript 异步的 Node 模块。下面代码演示怎样用 async.js 并行调用多个方法的:

**function** **runInParallel**() {

async.parallel([

getUserProfile,

getRecentActivity,

getSubscriptions,

getNotifications

], **function**(err, results) {

*//This callback runs when all the functions complete*

});

}

如果你想更深入了解 async.js ，请移步它的 GitHub 页面。

### 2. 异步

根据设计 Node.js 是单线程的。基于这点，同步代码会堵塞整个应用。比如说，多数的文件系统 API 都有它们的同步版本。下面代码演示了文件读取的同步和异步两种操作:

*// Asynchronous*

fs.readFile('file.txt', **function**(err, buffer) {

**var** content = buffer.toString();

});

*// Synchronous*

**var** content = fs.readFileSync('file.txt').toString();

不过要是你执行那种长时间的阻塞操作，主线程就会被阻塞到这些操作完成为止。这大大降低你应用的性能。所以，最好确保你的代码里用的都是异步版本 API，最起码你应该在性能节点异步。而且，你在选用第三方模块的时候也要很小心。因为当你想方设法把同步操作从你代码中剔除之后，一个外部库的同步调用会让你前功尽弃，降低你的应用性能。

### 3. 缓存

如果你用到一些不经常变化的数据，你应该把它们缓存起来，改善性能。比如说，下面的代码是获取最新帖子并显示的例子:

**var** router = express.Router();

router.route('/latestPosts').get(**function**(req, res) {

Post.getLatest(**function**(err, posts) {

**if** (err) {

**throw** err;

}

res.render('posts', { posts: posts });

});

});

如果你不经常发贴的话，你可以把帖子列表缓存起来，然后一段时间之后再把它们清理掉。比如，我们可以用 Redis 模块来达到这个目的。当然，你必须在你的服务器上装 Redis。然后你可以用叫做 node\_redis 的客户端来保存键/值对。下面的例子演示我们怎么缓存帖子:

**var** redis = require('redis'),

client = redis.createClient(null, null, { detect\_buffers: true }),

router = express.Router();

router.route('/latestPosts').get(**function**(req,res){

client.get('posts', **function** (err, posts) {

**if** (posts) {

**return** res.render('posts', { posts: JSON.parse(posts) });

}

Post.getLatest(**function**(err, posts) {

**if** (err) {

**throw** err;

}

client.set('posts', JSON.stringify(posts));

res.render('posts', { posts: posts });

});

});

});

看到了吧，我们首先检查 Redis 缓存，看看是否有帖子。如果有，我们从缓存中拿这些帖子列表。否则我们就检索数据库内容，然后把结果缓存。此外，一定时间之后，我们可以清理 Redis 缓存，这样就可以更新内容了。

### 4. gzip 压缩

开启 gzip 压缩对你的 Web 应用会产生巨大影响。当一个 gzip 压缩浏览器请求某些资源的时候，服务器会在响应返回给浏览器之前进行压缩。如果你不用 gzip 压缩你的静态资源，浏览器拿到它们可能会花费更长时间。在 Express 应用中，我们可以用内建 express.static() 中间件来处理静态内容。此外，还可以用 compression 中间件压缩和处理静态内容。下面是使用例:

**var** compression = **require**('compression');

app.**use**(compression()); *//use compression*

app.**use**(express.**static**(path.join(\_\_dirname, 'public')));

### 5. 如果可以，在用客户端渲染

现在有超多功能强劲的客户端 MVC/MVVM 框架，比如说 AngularJS, Ember, Meteor, 等等，构建一个单页面应用变得非常简单。基本上，你只要公开一个 API，返回 JSON 响应给客户端就可以了，而不需要在服务端渲染页面。在客户端，你可以用框架来组织 JSON 然后把它们显示在 UI 上。服务端只发送 JSON 响应可以节省带宽，改善性能，因为你不需要在每个响应里面都返回布局标记了，对吧，你只需要返回纯 JSON，然后在客户端渲染它们。看下我的这个教程，它是关于怎样用 Express 4 公开一个 RESTful APIs的。我还写了另一篇教程，演示了怎样把这些 APIs 和 AngularJS 结合起来。

### 6. 不要在 Sessions 存储太多数据

典型的 Express 页面应用， Session 数据默认是保存在内存中的。当你把太多数据保存在 Session 的时候，会导致服务器开销显著增大。所以，要么你切换到别的储存方式来保存 Session 数据，要么尽量减少存储在 Session 中的数据量。比如说，当用户登录到你的应用的时候，你可以只在 Session 中保存他们的 ID 而不是整个用户数据对象。还有，对于那些你能够从 id 拿到对象的查询，你应该会喜欢用 MongoDB 或者 Redis 来存储 session 数据。

### 7. 优化查询

假设你有个博客，你要在主页上显示最新帖子。你可能会通过 Mongoose 这样取数据:

Post.find().limit(10).exec(**function**(err, posts) {

*//send posts to client*

});

不过问题是 Mongoose 的 find() 方法会把对象的所有字段都查询出来，而许多字段在主页上并不要求。比如说，commentsis 保存的是特定帖子的回复。我们不需要显示文章回复，所以我们可以在查询的时候把它给剔除掉。这无疑会提高速度。可以像这样优化上面那条查询:

Post.find().limit(10).exclude('comments').exec(**function**(err, posts) {

*//send posts to client*

});

### 8. 用标准的 V8 方法

集合上的一些操作，比如 map，reduce，和 forEach 不一定支持所有浏览器。我们可以通过前台的库解决部分浏览器兼容性问题。但对于 Node.js，你要确切知道 Google 的 V8 JavaScript 引擎支持哪些操作。这样，你就可以在服务端直接用这些内建方法来操作集合了。

### 9. 在 Node 前面用 Nginx

Nginx 是个微小型轻量 Web 服务器，用它可以降低你的 Node.js 服务器的负载。你可以把静态资源配置到 nginx 上，而不是在 Node 上。你可以在 nginx 上用 gzip 压缩响应，让所有的响应都变得更小。所以，如果你有个正在营运的产品，我觉得你应该会想用 nginx 来改善运行速度的。

### 10. 打包 JavaScript

最后，你还可以大大提高页面应用速度，通过把多个 JS 文件打包。当浏览器在页面渲染中碰到 <script\> 元素的时候会被堵塞，直到拿到这个脚本才继续运行(除非设置了异步属性)。比如，如果你的页面有五个 JavaScript 文件，浏览器会发出五个独立的 HTTP 请求来获取他们。如果把这五个文件压缩打包成一个，整体性能将可以大幅提升。CSS 文件也是一样。你可以用诸如 Grunt/Gulp 这样的编译工具来打包你的资源文件。

**知识点**

布局 、 盒子模型 、 选择器优先级 、 CSS3 、 Flexbox 、 浮动元素 、Sass 、 Less ......

**题目&答案**

* 介绍一下CSS的盒子模型。
* (1)有两种， IE 盒子模型、标准 W3C 盒子模型；IE的content部分包含了 border 和 pading；

(2)盒模型： 内容(content)、填充(padding)、边界(margin)、 边框(border)。

* CSS选择符有哪些？哪些属性可以继承？
* (1)id选择器（**#classId）**
* (2)类选择器（.className）
* (3)标签选择器（div，h1，p，span，table）
* (4)相邻选择器（h1 + p）
* (5)子选择器（ul > li）
* (6)后代选择器（li a）
* (7)通配符选择器（\*）
* (8)属性选择器（a[rel = "external"]）
* (9)伪类选择器（a:hover，li:nth-child）
* *//*
* 可继承的样式：font-size，font-family，color，UL，LI，DL，DD，DT
* *//*

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

* CSS 如何计算优先级？
* 就近原则，同权重情况下以样式定义最近者为准；
* 载入样式以最后载入的定位为准；
* 优先级：
* !important > id > **class > tag**

**important 比 内联优先级高**

* CSS3 新增的伪类有哪些？
* CSS3新增伪类举例：
* p:first-of-type 选择属于其父元素的首个 <p> 元素的每个 <p> 元素。
* p:last-of-type 选择属于其父元素的最后 <p> 元素的每个 <p> 元素。
* p:only-of-type 选择属于其父元素唯一的 <p> 元素的每个 <p> 元素。
* p:only-child 选择属于其父元素的唯一子元素的每个 <p> 元素。
* 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。
* :enabled :disabled 控制表单控件的禁用状态。

:checked 单选框或复选框被选中。

* 如何居中div？如何居中一个浮动元素？如何让绝对定位的div居中？

*给 div 设置一个宽度，然后添加 margin: 0 auto 属性*

div{

width: 200px;

margin: 0 auto;

}

*居中一个浮动元素*

确定容器的宽高（500\*300）

设置层的外边距

**.element**{

width: 500px;

height: 300px; //高度可以不设

margin: -150px 0 0 -250px;

position: relative; //相对定位

background-color: red; //显示效果

left: 50%;

top: 50%;

}

*让绝对定位的 div 居中*

**.element**{

position: absolute;

width: 1200px;

background: none;

margin: 0 auto;

top: 0;

left: 0;

bottom: 0;

right: 0;

}

* display 有哪些值？各有何作用？
* b**lock** 象块类型元素一样显示。
* none 缺省值。象行内元素类型一样显示。
* inline-block 象行内元素一样显示，但其内容象块类型元素一样显示。

list-item 象块类型元素一样显示，并添加样式列表标记。

* position 中的 relative 和 absolute 的定位原点是什么？
* absolute
* 生成绝对定位的元素，相对于 **static** 定位以外的第一个父元素进行定位。
* *//*
* **fixed** （老IE不支持）
* 生成绝对定位的元素，相对于浏览器窗口进行定位。
* *//*
* relative
* 生成相对定位的元素，相对于其正常位置进行定位。
* *//*
* **static**
* 默认值。没有定位，元素出现在正常的流中
* （忽略 top, bottom, left, right z-index 声明）。
* *//*
* inherit

规定从父元素继承 position 属性的值。

* CSS3 有哪些新特性？
* 圆角-半径（border-radius:8px），
* 阴影（box-shadow:10px），
* 文字特效（text-shadow、），
* 线性渐变（gradient），
* 旋转（transform）
* transform:rotate(9deg) scale(0.85,0.90) translate(0px,-30px) skew(-9deg,0deg);*//旋转,缩放,定位,倾斜*
* 增加了更多的CSS选择器

多背景 rgba

* 解释一下 CSS3 的 Flexbox （弹性盒布局模型），以及适用场景？
* 用纯 CSS 创建一个三角形的原理是什么？
* 把上、左、右三条边隐藏掉（颜色设为 transparent）
* **#demo** {
* width:0;
* height: 0;
* border-width: 20px;
* border-style: solid;
* border-color: transparent transparent red transparent;

}

* 如何设计一个满屏“品”字布局？
* 简单的方式：
* 上面的div宽100%，
* 下面的两个div分别宽50%，

用float或inline使其不换行。

* 常见兼容性问题有哪些？
* (1)png24位的图片在iE6浏览器上出现背景，解决方案是做成PNG8。
* (2)浏览器默认的margin和padding不同。解决方案是加一个全局的\*{margin:0;padding:0;}来统一。
* (3)IE6双边距bug:块属性标签float后，又有横行的margin情况下，在ie6显示margin比设置的大。
* (4)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。
* (5)
* 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；
* 解决方法：（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。

(6)Chrome 中文界面下默认会将小于 12px 的文本强制按照 12px 显示,可通过加入 CSS 属性 -webkit-text-size-adjust: none; 解决。

* li与li之间有看不见的空白间隔是什么原因引起的？有什么解决办法？
* 经常遇到的浏览器的兼容性有哪些？原因，解决方法是什么，常用hack的技巧？
* 为什么要初始化CSS样式？
* 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。
* 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。
* //
* 最简单的初始化方法： \* {padding: 0; margin: 0;} （强烈不建议）
* //
* 淘宝的样式初始化代码：
* body, h1, h2, h3, h4, h5, h6, hr, p, blockquote, dl, dt, dd, ul, ol, li, pre, form, field**set**, 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; }
* field**set**, img { border:0; }
* button, input, **select**, textarea { font-**size**:100%; }

table { border-collapse:collapse; border-spacing:0; }

* absolute 的 containing block（容器块）计算方式跟正常流有什么不同？
* 无论属于哪种，都要先找到其祖先元素中最近的 position 值不为 static 的元素，然后再判断：
* (1)若此元素为 inline 元素，则 containing b**lock** 为能够包含这个元素生成的第一个和最后一个 inline box 的 padding box (除 margin, border 外的区域) 的最小矩形；
* (2)否则,则由这个祖先元素的 padding box 构成。
* 如果都找不到，则为 initial containing block。
* //
* 补充：
* (1)static(默认的)/**relative**：简单说就是它的父元素的内容框（即去掉padding的部分）
* (2)**absolute**: 向上找最近的定位为**absolute**/**relative**的元素

(3)fixed: 它的containing block一律为根元素(html/body)，根元素也是initial containing block

* CSS 里的 visibility 属性有个 collapse 属性值是干嘛用的？在不同浏览器下有什么区别？
* position 跟display、margin collapse、overflow、float这些特性相互叠加后会怎么样？
* 谈谈你对 BFC 规范（块级格式化上下文：block formatting context）的理解？
* （W3C CSS 2.1 规范中的一个概念,它是一个独立容器，决定了元素如何对其内容进行定位,以及与其他元素的关系和相互作用。）
* 一个页面是由很多个 Box 组成的,元素的类型和 display 属性,决定了这个 Box 的类型。

不同类型的 Box,会参与不同的 Formatting Context（决定如何渲染文档的容器）,因此Box内的元素会以不同的方式渲染,也就是说BFC内部的元素和外部的元素不会互相影响。

* CSS 定义的权重
* 以下是权重的规则：标签的权重为1，class的权重为10，id的权重为100，以下例子是演示各种定义的权重值：
* /\*权重为1\*/
* div{
* }
* /\*权重为10\*/
* .class1{
* }
* /\*权重为100\*/
* *#id1{*
* }
* /\*权重为100+1=101\*/
* *#id1 div{*
* }
* /\*权重为10+1=11\*/
* .class1 div{
* }
* /\*权重为10+10+1=21\*/
* .class1 .class2 div{
* }
* //

如果权重相同，则最后定义的样式会起作用，但是应该避免这种情况出现

* 请解释一下为什么会出现浮动？什么时候需要清除浮动？以及清除浮动的方式是什么？
* 移动端的布局用过媒体查询吗？
* CSS 优化、提高性能的方法有哪些？
* 浏览器是怎样解析 CSS 选择器的？
* 在网页中应该使用奇数还是偶数的字体？为什么？
* margin 和 padding 分别适合什么场景使用？
* 抽离样式模块怎么写？说出思路，有无实践经验？（阿里）
* 元素竖向的百分比设定是相对于容器的高度吗？
* 全屏滚动的原理是什么？用到了 CSS 的哪些属性？
* 什么是响应式设计？响应式设计的基本原理是什么？如何兼容低版本的IE？
* 视差滚动效果，如何给每页做不同的动画？（回到顶部，向下滑动要再次出现，和只出现一次分别怎么做？）
* ::before 和 :after 中双冒号和单冒号有什么区别？解释一下这2个伪元素的作用。
* 如何修改 Chrome 记住密码后自动填充表单的黄色背景？
* 你对 line-height 是如何理解的？
* 设置元素浮动后，该元素的 display 值是多少？（自动变成display:block）
* 怎么让 Chrome 支持小于12px的字符？
* 让页面里的字体变清晰、变细用 CSS 怎么做？（-webkit-font-smoothing: antialiased;）
* font-style 属性可以让它赋值为“oblique”，oblique 是什么意思？
* position: fixed; 在 Android 下无效怎么处理？
* 如果需要手动写动画，你认为最小时间间隔是多久？为什么？（阿里）

多数显示器默认频率是60Hz，即1秒刷新60次，所以理论上最小时间间隔为 1/6\*1000ms = 16.7ms

* overflow: scroll 时不能平滑滚动的问题怎么处理？
* png、jpg、gif 这些图片格式解释一下，分别什么时候用。有没有了解过webp、jpg-large？
* 什么是 Cookie 隔离？（或者说：请求资源的时候不要让它带cookie怎么做）
* 如果静态文件都放在主域名下，那静态文件请求的时候都带有的cookie的数据提交给server的，非常浪费流量，所以不如隔离开。
* 因为cookie有域的限制，因此不能跨域提交请求，故使用非主要域名的时候，请求头中就不会带有cookie数据，这样可以降低请求头的大小，降低请求时间，从而达到降低整体请求延时的目的。

同时这种方式不会将cookie传入Web Server，也减少了Web Server对cookie的处理分析环节，提高了webserver的http请求的解析速度。

* 介绍一下 Sass 和 Less 是什么？它们有何区别？
* Sass (Syntactically Awesome Stylesheets)是一种动态样式语言，语法跟css一样(但多了些功能)，比css好写，而且更容易阅读。Sass语法类似与Haml，属于缩排语法（makeup），用意就是为了快速写Html和Css。
* Less一种动态样式语言. 将CSS赋予了动态语言的特性，如变量，继承，运算， 函数. LESS 既可以在客户端上运行 (支持IE 6+, Webkit, Firefox)，也可一在服务端运行 (借助 Node.js)。
* 区别：
* (1))Sass是基于Ruby的，是在服务端处理的，而Less是需要引入less.js来处理Less代码输出Css到浏览器，也可以在开发环节使用Less，然后编译成Css文件，直接放到项目中，也有Less.app、SimpleLess、CodeKit.app这样的工具，也有在线编译地址。
* (2)变量符不一样，less是@，而Scss是$，而且变量的作用域也不一样，后面会讲到。
* (3)输出设置，Less没有输出设置，Sass提供4中输出选项：nested, compact, compressed 和 expanded。

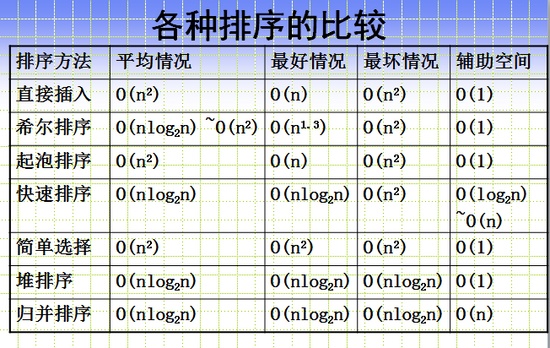
(4)Sass支持条件语句，可以使用**if**{}**else**{},**for**{}循环等等。而Less不支持。

## FE-interview

个人收集的前端知识点、面试题和答案，参考答案仅代表个人观点，方便复习。

## HTML， HTTP，web综合问题

### 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

1. 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
2. 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
3. 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
4. 重要内容不要用js输出：爬虫不会执行js获取内容
5. 少用iframe：搜索引擎不会抓取iframe中的内容
6. 非装饰性图片必须加alt
7. 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

1. cookie
2. session
3. url重写
4. 隐藏input
5. ip地址

### <img> 的 title 和 alt 有什么区别

1. title 是 ***[global attributes](http://www.w3.org/TR/html-markup/global-attributes.html" \l "common.attrs.core" \t "_blank)*** 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。
2. 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

1. <!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
2. <!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
3. 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。
4. 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
5. HTML5不基于SGML，所以不用指定DTD

#### 常见dotype：

1. **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">
2. **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">
3. **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">
4. **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">
5. **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">
6. **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">
7. **HTML 5** : <!doctype html>

### HTML全局属性(global attribute)有哪些

参考资料： ***[MDN: html global attribute](https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes" \t "_blank)*** 或者 ***[W3C HTML global-attributes](http://www.w3.org/TR/html-markup/global-attributes.html" \l "common.attrs.core" \t "_blank)***

* accesskey :设置快捷键，提供快速访问元素如 ***[aaa](https://github.com/vincenttesla/FE-interview" \t "_blank)*** 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](http://en.wikipedia.org/wiki/Microformats" \t "_blank)*** 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

1. 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
2. **GET** 是最常用的方法，通常用于 **请求服务器发送某个资源** 。
3. **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**
4. **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
5. **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。
6. **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。
7. **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。
8. **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)

1. 在浏览器地址栏输入URL
2. 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
3. 浏览器 **解析URL** 获取协议，主机，端口，path
4. 浏览器 **组装一个HTTP（GET）请求报文**
5. 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）
6. **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**
7. TCP链接建立后 **发送HTTP请求**
8. 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
9. 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
10. 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
11. 服务器将 **响应报文通过TCP连接发送回浏览器**
12. 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
13. 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
14. 如果资源可缓存， **进行缓存**
15. 对响应进行 **解码** （例如gzip压缩）
16. 根据资源类型决定如何处理（假设资源为HTML文档）
17. 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
18. **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
19. 解析过程中遇到图片、样式表、js文件， **启动下载**
20. 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
21. ***[根据DOM树和CSSOM树构建渲染树](https://developers.google.com/web/fundamentals/performance/critical-rendering-path/render-tree-construction" \t "_blank)***:
    * 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
22. **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 **difer** 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
23. **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](http://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html" \t "_blank)*** 中进行了定义：

1. 首行是 **Request-Line** 包括： **请求方法** ， **请求URI** ， **协议版本** ， **CRLF**
2. 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
3. 请求头和消息实体之间有一个 **CRLF分隔**
4. 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.htmlHTTP/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 (WindowsNT 6.1; WOW64) AppleWebKit/537.36 (KHTML, likeGecko) 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](http://www.w3.org/Protocols/rfc2616/rfc2616-sec6.html" \t "_blank)*** 中进行了定义：

1. 首行是状态行包括： **HTTP版本，状态码，状态描述** ，后面跟一个CRLF
2. 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**
3. 响应头部和响应实体之间用 **一个CRLF空行** 分隔
4. 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; char**set**=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](https://developer.yahoo.com/performance/rules.html" \t "_blank)*** ：

* 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

1. 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
2. Cookie方面
   * 减小cookie大小
   * 引入资源的域名不要包含cookie
3. 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
4. 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
5. 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
6. 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](http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html" \t "_blank)***

* 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。

1. 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** ：
2. 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** ：
3. 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** :
4. 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

1. \*\*通用选择器 ：选择所有元素， 不参与计算优先级 \*，兼容性IE6+
2. **#X id选择器** ：选择id值为X的元素，兼容性：IE6+
3. **.X 类选择器** ： 选择class包含X的元素，兼容性：IE6+
4. **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+
5. **X 元素选择器** ： 选择标所有签为X的元素，兼容性：IE6+
6. :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+
7. **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+
8. **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+
9. **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+
10. **[attr]** ：选择所有设置了attr属性的元素，兼容性IE7+
11. **[attr=value]** ：选择属性值刚好为value的元素
12. **[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
13. **[attr|=value]** ：选择属性值刚好为value或者value-开头的元素
14. **[attr^=value]** ：选择属性值以value开头的元素
15. **[attr$=value]** ：选择属性值以value结尾的元素
16. **[attr\*=value]** ：选择属性值中包含value的元素
17. **[: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+
18. **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+
19. **:hover** ：鼠标移入状态的元素，兼容性a标签IE4+， 所有元素IE7+
20. **:not(selector)** ：选择不符合selector的元素。 **不参与计算优先级** ，兼容性：IE9+
21. **::first-letter** ：伪元素，选择块元素第一行的第一个字母，兼容性IE5.5+
22. **::first-line** ：伪元素，选择块元素的第一行，兼容性IE5.5+
23. **: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+
24. **: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+
25. **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+
26. **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+
27. **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+
28. **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+
29. **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+
30. **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+
31. **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：

1. 减少HTTP请求数，极大地提高页面加载速度
2. 增加图片信息重复度，提高压缩比，减少图片大小
3. 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：

1. 图片合并麻烦
2. 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：

1. 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
2. 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
3. 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。
4. 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]>*

*Specialinstructionsfor 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 \*\*\*\*\*\*/

/\* IE6 and below \*/

\* html *#uno  { color: red }*

/\* IE7 \*/

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

/\* IE7, FF, Saf, Opera  \*/

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

/\* IE8, FF, Saf, Opera (Everything but IE 6,7) \*/

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

/\* Opera 9.27 **and** below, safari 2 \*/

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

/\* Safari 2-3 \*/

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

/\* safari 3+, chrome 1+, opera9+, ff 3.5+ \*/

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

/\* safari 3+, chrome 1+, opera9+, ff 3.5+ \*/

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

/\* saf3+, chrome1+ \*/

@mediascreenand (-webkit-min-device-pixel-ratio:0) {

*#diez  { color: red  }*

}

/\* iPhone / mobile webkit \*/

@mediascreenand (max-device-width: 480px) {

*#veintiseis { color: red  }*

}

/\* Safari 2 - 3.1 \*/

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

/\* Safari 2 - 3.1, Opera 9.25 \*/

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

/\* Everything but IE6-8 \*/

:root \*> *#quince { color: red  }*

/\* IE7 \*/

\*+html *#dieciocho {  color: red }*

/\* Firefox only. 1+ \*/

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

/\* Firefox 3.0+ \*/

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

* 属性hack：不同浏览器解析bug或方法

*/\* IE6 \*/*

#once { \_color: blue }

*/\* IE6, IE7 \*/*

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

*/\* Everything but IE6 \*/*

#diecisiete { color*/\*\*/: blue }*

*/\* IE6, IE7, IE8 \*/*

#diecinueve { color: blue\9; }

*/\* IE7, IE8 \*/*

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

*/\* IE6, IE7 -- acts as an !important \*/*

#veintesiete { color: blue !ie; } */\* string after ! can be anything \*/*

### 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

1. 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**
2. 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

1. link 是HTML方式， @import 是CSS方式
2. 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](http://www.bluerobot.com/web/css/fouc.asp/" \t "_blank)***
3. link 可以通过 rel="alternate stylesheet" 指定候选样式
4. 浏览器对 link 支持早于 @import ，可以使用 @import 对老浏览器隐藏样式
5. @import 必须在样式规则之前，可以在css文件中引用其他文件
6. 总体来说： ***[link优于@import](http://www.stevesouders.com/blog/2009/04/09/dont-use-import/" \t "_blank)***

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

#### block元素特点：

* 处于常规流中时，如果 width 没有设置，会自动填充满父容器
* 可以应用 margin/padding
* 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
* 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）
* 忽略 vertical-align

#### inline 元素特点

* 水平方向上根据 direction 依次布局
* 不会在元素前后进行换行
* 受 white-space 控制
* margin/padding 在竖直方向上无效，水平方向上有效
* width/height 属性对非替换行内元素无效，宽度由元素内容决定
* 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
* 浮动或绝对定位时会转换为 block
* 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](http://www.yuiblog.com/blog/2008/11/04/imageopt-2/" \t "_blank)***

#### GIF:

1. 8位像素，256色
2. 无损压缩
3. 支持简单动画
4. 支持boolean透明
5. 适合简单动画

#### JPEG：

1. 颜色限于256
2. 有损压缩
3. 可控制压缩质量
4. 不支持透明
5. 适合照片

#### PNG：

1. 有PNG8和truecolor PNG
2. PNG8类似GIF颜色上限为256，文件小，支持alpha透明度，无动画
3. 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](https://developer.mozilla.org/en-US/docs/Web/CSS/font" \t "_blank)***
  + ***[word-break](https://developer.mozilla.org/en-US/docs/Web/CSS/word-break" \t "_blank)***
  + ***[letter-spacing](https://developer.mozilla.org/en-US/docs/Web/CSS/letter-spacing" \t "_blank)***
  + ***[text-align](https://developer.mozilla.org/en-US/docs/Web/CSS/text-align" \t "_blank)***
  + ***[text-rendering](https://developer.mozilla.org/en-US/docs/Web/CSS/text-rendering" \t "_blank)***
  + ***[word-spacing](https://developer.mozilla.org/en-US/docs/Web/CSS/word-spacing" \t "_blank)***
  + ***[white-space](https://developer.mozilla.org/en-US/docs/Web/CSS/white-space" \t "_blank)***
  + ***[text-indent](https://developer.mozilla.org/en-US/docs/Web/CSS/text-indent" \t "_blank)***
  + ***[text-transform](https://developer.mozilla.org/en-US/docs/Web/CSS/text-transform" \t "_blank)***
  + ***[text-shadow](https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow" \t "_blank)***

1. ***[line-height](https://developer.mozilla.org/en-US/docs/Web/CSS/line-height" \t "_blank)***
2. ***[color](https://developer.mozilla.org/en-US/docs/Web/CSS/color" \t "_blank)***
3. ***[visibility](https://developer.mozilla.org/en-US/docs/Web/CSS/visibility" \t "_blank)***
4. ***[cursor](https://developer.mozilla.org/en-US/docs/Web/CSS/cursor" \t "_blank)***

### 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;

<styletype="text/css">

.outer{

    width: 215px;

    height: 100px;

    border: 1px solid red;

    overflow: auto;

    position: relative;  /\* 修复bug \*/

}

.inner{

    width: 100px;

    height: 200px;

    background-color: purple;

    position: relative;

}

</style>

<divclass="outer">

    <divclass="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：

<styletype="text/css">

.p:hover,

.hover{

    background: purple;

}

</style>

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

<scripttype="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); */\* for IE5-7 \*/*

    -ms-filter: "progid:DXImageTransform.Microsoft.Alpha(Opacity=60)"; */\* for IE 8\*/*

}

* 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-b**lock**;

    \*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

1. 容器元素闭合标签前添加额外元素并设置 clear: both
2. 父元素触发块级格式化上下文(见块级可视化上下文部分)
3. 设置容器元素伪元素进行清理 ***[推荐的清理浮动方法](http://nicolasgallagher.com/micro-clearfix-hack/" \t "_blank)***

*/\*\**

*\* 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: " "; */\* 1 \*/*

    display: table; */\* 2 \*/*

}

**.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：

1. 根元素
2. 浮动元素（ float 不是 none ）
3. 绝对定位元素（ position 取值为 absolute 或 fixed ）
4. display 取值为 inline-block , table-cell , table-caption ,flex , inline-flex 之一的元素
5. overflow 不是 visible 的元素

#### 作用：

1. 可以包含浮动元素
2. 不被浮动元素覆盖
3. 阻止父子元素的margin折叠

### display,float,position的关系

1. 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
2. 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。
3. 否则，如果float不是none，框是浮动的，display根据下表进行调整
4. 否则，如果元素是根元素，display根据下表进行调整
5. 其他情况下display的值为指定值

#### 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：

1. 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
2. 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
3. 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
4. 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)

1. 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。
2. 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
3. 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
4. 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
5. 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）：

1. 根元素的边界和背景
2. 常规流中的元素按照html中顺序
3. 浮动块
4. positioned元素按照html中出现顺序

如何创建stacking context：

1. 根元素
2. z-index不为auto的定位元素
3. a flex item with a z-index value other than ‘auto’
4. opacity小于1的元素
5. 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>

    <divclass="content">

    aaaaaa aaaaaa a a a a a a a a

    </div>

</body>

<style>

    body{

        background: #DDD;

        text-align: center; */\* 3 \*/*

    }

**.content**{

        width: 500px;      */\* 1 \*/*

        text-align: left;  */\* 3 \*/*

        margin: 0 auto;    */\* 2 \*/*

        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>

    <divclass="content">

    aaaaaa aaaaaa a a a a a a a a

    </div>

</body>

<style>

    body{

        background: #DDD;

    }

**.content**{

        width: 500px;        */\* 1 \*/*

        float: left;

        position: relative;  */\* 2 \*/*

        left: 50%;            */\* 3 \*/*

        margin-left: -250px;  */\* 4 \*/*

        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>

    <divclass="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>

    <divclass="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](http://www.vanseodesign.com/css/vertical-centering/" \t "_blank)*** 。 ***[盘点8种CSS实现垂直居中](http://blog.csdn.net/freshlover/article/details/11579669" \t "_blank)***

* 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](http://msdn.microsoft.com/en-us/library/ie/hh781509(v=vs.85).aspx" \t "_blank)***

### XMLHttpRequest通用属性和方法

1. 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
2. onreadystatechange ：readyState改变时调用的函数
3. status ：服务器返回的HTTP状态码（如，200， 404）
4. statusText :服务器返回的HTTP状态信息（如，OK，No Content）
5. responseText :作为字符串形式的来自服务器的完整响应
6. responseXML : Document对象，表示服务器的响应解析成的XML文档
7. abort() :取消异步HTTP请求
8. 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
9. getResponseHeader(headerName) :返回headName对应的报头值
10. 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
11. setRequestHeader(name, value) :设置HTTP报头
12. 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

1. focus/blur不冒泡，focusin/focusout冒泡
2. 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)
3. 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

1. 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
2. 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
3. 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：

<divid="target"><span>test</span></div>

<scripttype="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

1. 都会在浏览器端保存，有大小限制，同源限制
2. 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
3. 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
4. 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
5. 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
6. localStorage的修改会促发其他文档窗口的update事件
7. cookie有secure属性要求HTTPS传输
8. 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

1. 协议相同
2. 域名相同
3. 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](https://developer.mozilla.org/en-US/docs/Glossary/Symbol" \t "_blank)*** (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：

1. 函数本身作用域。
2. 闭包定义时的作用域。
3. 全局作用域。

闭包常见用途：

1. 创建特权方法用于访问控制
2. 事件处理程序及回调

### javascript有哪几种方法定义函数

1. ***[函数声明表达式](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/function" \t "_blank)***
2. ***[function操作符](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/function" \t "_blank)***
3. ***[Function 构造函数](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Function" \t "_blank)***
4. ***[ES6:arrow function](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/arrow_functions" \t "_blank)***

重要参考资料： ***[MDN:Functions\_and\_function\_scope](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions_and_function_scope" \t "_blank)***

### 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。

* 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
* 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
* 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：

CACHEMANIFEST

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](https://github.com/qiu-deqing/google/blob/master/module/js/cookieUtil.js" \t "_blank)*** ：自己写的cookie操作工具

### javascript有哪些方法定义对象

1. 对象字面量： var obj = {};
2. 构造函数： var obj = new Object();
3. Object.create(): var obj = Object.create(Object.prototype);

### ===运算符判断相等的流程是怎样的

1. 如果两个值不是相同类型，它们不相等
2. 如果两个值都是null或者都是undefined，它们相等
3. 如果两个值都是布尔类型true或者都是false，它们相等
4. 如果其中有一个是 **NaN** ，它们不相等
5. 如果都是数值型并且数值相等，他们相等， -0等于0
6. 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
7. 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

1. 如果两个值类型相同，按照===比较方法进行比较
2. 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

1. 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
2. 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
3. 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

1. 如果有操作数是对象，转换为原始值
2. 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
3. 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：

1. 优点：正确设置原型链实现继承
2. 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
3. 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
4. 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
5. 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：

1. 优点：正确设置原型链实现继承
2. 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：

1. 优点：正确设置原型链且避免方法1.2中的缺点
2. 缺点：ES5方法需要注意兼容性

#### 改进：

1. 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**);

}

1. 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
2. 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>

<divid="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** keyin obj) {

**if** (key.indexOf('post') === 0 && typeofobj[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');

### 完成下面的tool-tip

### 编写javascript深度克隆函数deepClone

**function** **deepClone**(obj) {

**var** \_toString = Object.prototype.toString;

*// null, undefined, non-object, function*

**if** (!obj || typeofobj !== 'object') {

**return** obj;

    }

*// DOM Node*

**if** (obj.nodeType && 'cloneNode' **in** obj) {

**return** obj.cloneNode(true);

    }

*// Date*

**if** (\_toString.call(obj) === '[object Date]') {

**return** **new** Date(obj.getTime());

    }

*// RegExp*

**if** (\_toString.call(obj) === '[object RegExp]') {

**var** flags = [];

**if** (obj.global) { flags.push('g'); }

**if** (obj.multiline) { flags.push('m'); }

**if** (obj.ignoreCase) { flags.push('i'); }

**return** **new** RegExp(obj.source, flags.join(''));

    }

**var** result = Array.isArray(obj) ? [] :

        obj.constructor ? **new** obj.constructor() : {};

**for** (**var** keyin obj ) {

        result[key] = deepClone(obj[key]);

    }

**return** result;

}

**function** **A**() {

**this**.a = a;

}

**var** a = {

    name: 'qiu',

    birth: **new** Date(),

    pattern: /qiu/gim,

    container: document.body,

    hobbys: ['book', **new** Date(), /aaa/gim, 111]

};

**var** c = **new** A();

**var** b = deepClone(c);

console.log(c.a === b.a);

console.log(c, b);

### 补充代码,鼠标单击Button1后将Button1移动到Button2的后面

**<!doctypehtml>**

<html>

<head>

    <metacharset="utf-8">

    <title>TEst</title>

</head>

<body>

<div>

  <inputtype="button" id ="button1" value="1" />

  <inputtype="button" id ="button2" value="2" />

</div>

<scripttype="text/javascript">

    var btn1 = document.getElementById('button1');

    var btn2 = document.getElementById('button2');

    addListener(btn1, 'click', function (event) {

        btn1.parentNode.insertBefore(btn2, btn1);

    });

    function addListener(elem, type, handler) {

        if (elem.addEventListener) {

            elem.addEventListener(type, handler, false);

            return handler;

        } else if (elem.attachEvent) {

            function wrapper() {

                var event = window.event;

                event.target = event.srcElement;

                handler.call(elem, event);

            }

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

            return wrapper;

        }

    }

</script>

</body>

</html>

### 网页中实现一个计算当年还剩多少时间的倒数计时程序,要求网页上实时动态显示”××年还剩××天××时××分××秒”

**<!doctypehtml>**

<html>

<head>

    <metacharset="utf-8">

    <title>TEst</title>

</head>

<body>

    <spanid="target"></span>

<scripttype="text/javascript">

    // 为了简化。每月默认30天

    function getTimeString() {

        var start = new Date();

        var end = new Date(start.getFullYear() + 1, 0, 1);

        var elapse = Math.floor((end - start) / 1000);

        var seconds = elapse % 60 ;

        var minutes = Math.floor(elapse / 60) % 60;

        var hours = Math.floor(elapse / (60 \* 60)) % 24;

        var days = Math.floor(elapse / (60 \* 60 \* 24)) % 30;

        var months = Math.floor(elapse / (60 \* 60 \* 24 \* 30)) % 12;

        var years = Math.floor(elapse / (60 \* 60 \* 24 \* 30 \* 12));

        return start.getFullYear() + '年还剩' + years + '年' + months + '月' + days + '日'

            + hours + '小时' + minutes + '分' + seconds + '秒';

    }

    function domText(elem, text) {

        if (text == undefined) {

            if (elem.textContent) {

                return elem.textContent;

            } else if (elem.innerText) {

                return elem.innerText;

            }

        } else {

            if (elem.textContent) {

                elem.textContent = text;

            } else if (elem.innerText) {

                elem.innerText = text;

            } else {

                elem.innerHTML = text;

            }

        }

    }

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

    setInterval(function () {

        domText(target, getTimeString());

    }, 1000)

</script>

</body>

</html>

### 完成一个函数,接受数组作为参数,数组元素为整数或者数组,数组元素包含整数或数组,函数返回扁平化后的数组

如：[1, [2, [ [3, 4], 5], 6]] => [1, 2, 3, 4, 5, 6]

**var** data =  [1, [2, [ [3, 4], 5], 6]];

**function** **flat**(data, result) {

**var** i, d, len;

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

            d = data[i];

**if** (**typeof** d === 'number') {

                result.push(d);

            } **else** {

                flat(d, result);

            }

        }

    }

**var** result = [];

    flat(data, result);

    console.log(result);

### 如何判断一个对象是否为数组

如果浏览器支持Array.isArray()可以直接判断否则需进行必要判断

*/\*\**

*\* 判断一个对象是否是数组，参数不是对象或者不是数组，返回false*

*\**

*\* @param {Object} arg 需要测试是否为数组的对象*

*\* @return {Boolean} 传入参数是数组返回true，否则返回false*

*\*/*

function isArray(arg) {

**if** (typeofarg === 'object') {

**return** Object.prototype.toString.call(arg) === '[object Array]';

    }

**return** **false**;

}

### 请评价以下事件监听器代码并给出改进意见

**if** (window.addEventListener) {

**var** addListener = **function** (el, type, listener, useCapture) {

    el.addEventListener(type, listener, useCapture);

  };

}

**else** **if** (document.all) {

  addListener = **function** (el, type, listener) {

    el.attachEvent('on' + type, **function** () {

      listener.apply(el);

    });

  };

}

作用：浏览器功能检测实现跨浏览器DOM事件绑定

#### 优点：

1. 测试代码只运行一次，根据浏览器确定绑定方法
2. 通过 listener.apply(el) 解决IE下监听器this与标准不一致的地方
3. 在浏览器不支持的情况下提供简单的功能，在标准浏览器中提供捕获功能

#### 缺点：

1. document.all作为IE检测不可靠，应该使用if(el.attachEvent)
2. addListener在不同浏览器下API不一样
3. listener.apply 使this与标准一致但监听器无法移除
4. 未解决IE下listener参数event。 target问题

#### 改进:

**var** addListener;

**if** (window.addEventListener) {

  addListener = **function** (el, type, listener, useCapture) {

    el.addEventListener(type, listener, useCapture);

**return** listener;

  };

}

**else** **if** (window.attachEvent) {

  addListener = **function** (el, type, listener) {

*// 标准化this，event，target*

**var** wrapper = **function** () {

**var** event = window.event;

      event.target = event.srcElement;

      listener.call(el, event);

    };

    el.attachEvent('on' + type, wrapper);

**return** wrapper;

*// 返回wrapper。调用者可以保存，以后remove*

  };

}

### 如何判断一个对象是否为函数

*/\*\**

*\* 判断对象是否为函数，如果当前运行环境对可调用对象（如正则表达式）*

*\* 的typeof返回'function'，采用通用方法，否则采用优化方法*

*\**

*\* @param {Any} arg 需要检测是否为函数的对象*

*\* @return {boolean} 如果参数是函数，返回true，否则false*

*\*/*

function isFunction(arg) {

**if** (arg) {

**if** (typeof (/./) !== 'function') {

**return** typeofarg === 'function';

        } **else** {

**return** Object.prototype.toString.call(arg) === '[object Function]';

        }

    } *// end if*

**return** **false**;

}

### 编写一个函数接受url中query string为参数,返回解析后的Object,query string使用application/x-www-form-urlencoded编码

*/\*\**

*\* 解析query string转换为对象，一个key有多个值时生成数组*

*\**

*\* @param {String} query 需要解析的query字符串，开头可以是?，*

*\* 按照application/x-www-form-urlencoded编码*

*\* @return {Object} 参数解析后的对象*

*\*/*

function parseQuery(query) {

    var result = {};

*// 如果不是字符串返回空对象*

**if** (typeofquery !== 'string') {

**return** result;

    }

*// 去掉字符串开头可能带的?*

**if** (query.charAt(0) === '?') {

        query = query.substring(1);

    }

    var pairs = query.split('&');

    var pair;

    var key, value;

    var i, len;

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

        pair = pairs[i].split('=');

*// application/x-www-form-urlencoded编码会将' '转换为+*

        key = decodeURIComponent(pair[0]).replace(/\+/g, ' ');

        value = decodeURIComponent(pair[1]).replace(/\+/g, ' ');

*// 如果是新key，直接添加*

**if** (!(keyin result)) {

            result[key] = value;

        }

*// 如果key已经出现一次以上，直接向数组添加value*

**else** **if** (isArray(result[key])) {

            result[key].push(value);

        }

*// key第二次出现，将结果改为数组*

**else** {

            var arr = [result[key]];

            arr.push(value);

            result[key] = arr;

        } *// end if-else*

    } *// end for*

**return** result;

}

function isArray(arg) {

**if** (arg && typeofarg === 'object') {

**return** Object.prototype.toString.call(arg) === '[object Array]';

    }

**return** **false**;

}

*/\*\**

*console.log(parseQuery('sourceid=chrome-instant&ion=1&espv=2&ie=UTF-8'));*

*\*/*

### 解析一个完整的url,返回Object包含域与window.location相同

*/\*\**

*\* 解析一个url并生成window.location对象中包含的域*

*\* location:*

*\* {*

*\*      href: '包含完整的url',*

*\*      origin: '包含协议到pathname之前的内容',*

*\*      protocol: 'url使用的协议，包含末尾的:',*

*\*      username: '用户名', // 暂时不支持*

*\*      password: '密码',  // 暂时不支持*

*\*      host: '完整主机名，包含:和端口',*

*\*      hostname: '主机名，不包含端口'*

*\*      port: '端口号',*

*\*      pathname: '服务器上访问资源的路径/开头',*

*\*      search: 'query string，?开头',*

*\*      hash: '#开头的fragment identifier'*

*\* }*

*\**

*\* @param {string} url 需要解析的url*

*\* @return {Object} 包含url信息的对象*

*\*/*

**function** **parseUrl**(url) {

**var** result = {};

**var** keys = ['href', 'origin', 'protocol', 'host',

                'hostname', 'port', 'pathname', 'search', 'hash'];

**var** i, len;

**var** regexp = /(([^:]+:)\/\/(([^:\/\?#]+)(:\d+)?))(\/[^?#]\*)?(\?[^#]\*)?(#.\*)?/;

**var** match = regexp.exec(url);

**if** (match) {

**for** (i = keys.length - 1; i >= 0; --i) {

            result[keys[i]] = match[i] ? match[i] : '';

        }

    }

**return** result;

}

### 完成函数getViewportSize返回指定窗口的视口尺寸

*/\*\**

*\* 查询指定窗口的视口尺寸，如果不指定窗口，查询当前窗口尺寸*

*\*\*/*

function getViewportSize(w) {

    w = w || window;

*// IE9及标准浏览器中可使用此标准方法*

**if** ('innerHeight' in w) {

**return** {

            width: w.innerWidth,

            height: w.innerHeight

        };

    }

    var d = w.document;

*// IE 8及以下浏览器在标准模式下*

**if** (document.compatMode === 'CSS1Compat') {

**return** {

            width: d.documentElement.clientWidth,

            height: d.documentElement.clientHeight

        };

    }

*// IE8及以下浏览器在怪癖模式下*

**return** {

        width: d.body.clientWidth,

        height: d.body.clientHeight

    };

}

### 完成函数getScrollOffset返回窗口滚动条偏移量

*/\*\**

*\* 获取指定window中滚动条的偏移量，如未指定则获取当前window*

*\* 滚动条偏移量*

*\**

*\* @param {window} w 需要获取滚动条偏移量的窗口*

*\* @return {Object} obj.x为水平滚动条偏移量,obj.y为竖直滚动条偏移量*

*\*/*

function getScrollOffset(w) {

    w =  w || window;

*// 如果是标准浏览器*

**if** (w.pageXOffset != **null**) {

**return** {

            x: w.pageXOffset,

            y: w.pageYOffset

        };

    }

*// 老版本IE，根据兼容性不同访问不同元素*

    var d = w.document;

**if** (d.compatMode === 'CSS1Compat') {

**return** {

            x: d.documentElement.scrollLeft,

            y: d.documentElement.scrollTop

        }

    }

**return** {

        x: d.body.scrollLeft,

        y: d.body.scrollTop

    };

}

### 现有一个字符串richText,是一段富文本,需要显示在页面上.有个要求,需要给其中只包含一个img元素的p标签增加一个叫pic的class.请编写代码实现.可以使用jQuery或KISSY.

**function** **richText**(text) {

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

    div.innerHTML = text;

**var** p = div.getElementsByTagName('p');

**var** i, len;

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

**if** (p[i].getElementsByTagName('img').length === 1) {

            p[i].classList.add('pic');

        }

    }

**return** div.innerHTML;

}

### 请实现一个Event类,继承自此类的对象都会拥有两个方法on,off,once和trigger

**function** **Event**() {

**if** (!(**this** **instanceof** Event)) {

**return** **new** Event();

    }

**this**.\_callbacks = {};

}

Event.prototype.on = **function** (type, handler) {

    this\_callbacks = **this**.\_callbacks || {};

**this**.\_callbacks[type] = **this**.callbacks[type] || [];

**this**.\_callbacks[type].push(handler);

**return** **this**;

};

Event.prototype.off = **function** (type, handler) {

**var** **list** = **this**.\_callbacks[type];

**if** (**list**) {

**for** (**var** i = **list**.length; i >= 0; --i) {

**if** (**list**[i] === handler) {

**list**.splice(i, 1);

            }

        }

    }

**return** **this**;

};

Event.prototype.trigger = **function** (type, data) {

**var** **list** = **this**.\_callbacks[type];

**if** (**list**) {

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

**list**[i].call(**this**, data);

        }

    }

};

Event.prototype.once = **function** (type, handler) {

**var** **self** = **this**;

**function** **wrapper**() {

        handler.apply(**self**, arguments);

**self**.off(type, wrapper);

    }

**this**.on(type, wrapper);

**return** **this**;

};

### 编写一个函数将列表子元素顺序反转

<ulid="target">

    <li>1</li>

    <li>2</li>

    <li>3</li>

    <li>4</li>

</ul>

<script>

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

**var** i;

**var** frag = document.createDocumentFragment();

**for** (i = target.children.length - 1; i >= 0; --i) {

        frag.appendChild(target.children[i]);

    }

    target.appendChild(frag);

</script>

### 以下函数的作用是?空白区域应该填写什么

*// define*

(**function** (window) {

**function** **fn**(str) {

**this**.str = str;

    }

    fn.prototype.format = **function** () {

**var** arg = \_\_1\_\_;

**return** **this**.str.replace(\_\_2\_\_, **function** (a, b) {

**return** arg[b] || '';

        });

    };

    window.fn = fn;

})(window);

*// use*

(**function** () {

**var** t = **new** fn('<p><a href="{0}">{1}</a><span>{2}</span></p>');

    console.log(t.format('http://www.alibaba.com', 'Alibaba', 'Welcome'));

})();

define部分定义一个简单的模板类，使用{}作为转义标记，中间的数字表示替换目标，format实参用来替换模板内标记横线处填：

1. Array.prototype.slice.call(arguments, 0)
2. /\{\s\*(\d+)\s\*\}/g

### 编写一个函数实现form的序列化(即将一个表单中的键值序列化为可提交的字符串)

<formid="target">

    <selectname="age">

        <optionvalue="aaa">aaa</option>

        <optionvalue="bbb" selected>bbb</option>

    </select>

    <selectname="friends" multiple>

        <optionvalue="qiu" selected>qiu</option>

        <optionvalue="de">de</option>

        <optionvalue="qing" selected>qing</option>

    </select>

    <inputname="name" value="qiudeqing">

    <inputtype="password" name="password" value="11111">

    <inputtype="hidden" name="salery" value="3333">

    <textareaname="description">description</textarea>

    <inputtype="checkbox" name="hobby" checkedvalue="football">Football

    <inputtype="checkbox" name="hobby" value="basketball">Basketball

    <inputtype="radio" name="sex" checkedvalue="Female">Female

    <inputtype="radio" name="sex" value="Male">Male

</form>

<script>

*/\*\**

*\* 将一个表单元素序列化为可提交的字符串*

*\**

*\* @param {FormElement} form 需要序列化的表单元素*

*\* @return {string} 表单序列化后的字符串*

*\*/*

**function** serializeForm(form) {

**if** (!form || form.nodeName.toUpperCase() !== 'FORM') {

**return**;

  }

**var** result = [];

**var** i, len;

**var** field, fieldName, fieldType;

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

    field = form.elements[i];

    fieldName = field.name;

    fieldType = field.type;

**if** (field.disabled || !fieldName) {

**continue**;

    } *// enf if*

**switch** (fieldType) {

**case** 'text':

**case** 'password':

**case** 'hidden':

**case** 'textarea':

        result.push(encodeURIComponent(fieldName) + '=' +

            encodeURIComponent(field.value));

**break**;

**case** 'radio':

**case** 'checkbox':

**if** (field.checked) {

          result.push(encodeURIComponent(fieldName) + '=' +

            encodeURIComponent(field.value));

        }

**break**;

**case** 'select-one':

**case** 'select-multiple':

**for** (**var** j = 0, jLen = field.options.length; j < jLen; ++j) {

**if** (field.options[j].selected) {

            result.push(encodeURIComponent(fieldName) + '=' +

              encodeURIComponent(field.options[j].value || field.options[j].text));

          }

        } *// end for*

**break**;

**case** 'file':

**case** 'submit':

**break**; *// 是否处理？*

**default**:

**break**;

    } *// end switch*

  } *// end for*

**return** result.join('&');

}

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

console.log(serializeForm(form));

</script>

### 使用原生javascript给下面列表中的li节点绑定点击事件,点击时创建一个Object对象,兼容IE和标准浏览器

<ulid="nav">

    <li><a href="http://11111">111</a></li>

    <li><a href="http://2222">222</a></li>

    <li><a href="http://333">333</a></li>

    <li><a href="http://444">444</a></li>

</ul>

Object:

{

    "index": 1,

    "name": "111",

    "link": "http://1111"

}

script:

**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) {

**function** **wrapper**(event) {

**return** 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;

        }

    }

};

**var** DOMUtil = {

    text: **function** (elem) {

**if** ('textContent' **in** elem) {

**return** elem.textContent;

        } **else** **if** ('innerText' **in** elem) {

**return** elem.innerText;

        }

    },

    prop: **function** (elem, propName) {

**return** elem.getAttribute(propName);

    }

};

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

EventUtil.on(nav, 'click', **function** (event) {

**var** event = EventUtil.getEvent(event);

**var** target = EventUtil.getTarget(event);

**var** children = **this**.children;

**var** i, len;

**var** anchor;

**var** obj = {};

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

**if** (children[i] === target) {

            obj.index = i + 1;

            anchor = target.getElementsByTagName('a')[0];

            obj.name = DOMUtil.text(anchor);

            obj.link = DOMUtil.prop(anchor, 'href');

        }

    }

    alert('index: ' + obj.index + ' name: ' + obj.name +

        ' link: ' + obj.link);

});

### 有一个大数组,var a = [‘1′, ‘2’, ‘3’, …];a的长度是100,内容填充随机整数的字符串.请先构造此数组a,然后设计一个算法将其内容去重

*/\*\**

*\* 数组去重*

*\*\*/*

**function** **normalize**(arr) {

**if** (arr && Array.isArray(arr)) {

**var** i, len, map = {};

**for** (i = arr.length; i >= 0; --i) {

**if** (arr[i] **in** map) {

                    arr.splice(i, 1);

                } **else** {

                    map[arr[i]] = true;

                }

            }

        }

**return** arr;

    }

*/\*\**

*\* 用100个随机整数对应的字符串填充数组。*

*\*\*/*

**function** **fillArray**(arr, start, end) {

        start = start == undefined ? 1 : start;

        end = end == undefined ?  100 : end;

**if** (end <= start) {

            end = start + 100;

        }

**var** width = end - start;

**var** i;

**for** (i = 100; i >= 1; --i) {

            arr.push('' + (Math.floor(Math.random() \* width) + start));

        }

**return** arr;

    }

**var** input = [];

    fillArray(input, 1, 100);

    input.sort(**function** (a, b) {

**return** a - b;

    });

    console.log(input);

    normalize(input);

    console.log(input);

资料刚刚收集，覆盖面比较广，包括了web端、移动端的知识点，根据你自己的需要选择性阅读即可。

再次提一下： 看 或 背 面试题不像学校考试，死记硬背是没什么用的，看面试题是对理论知识的总结让自己表达的时候知道怎么说。

答案有些不够正确和全面，问题问的不够好，欢迎补充你所知道的答案、技巧、题目；最好是现在网上找不到的。

格式不断修改更新中。

HTML、CSS部分

要点：对Web标准的理解、浏览器差异、CSS基本功：布局、盒子模型、选择器优先级及使用、HTML5、CSS3、移动端开发 技术等

1.Doctype作用? 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?

（1）、<!DOCTYPE> 声明位于文档中的最前面，处于 <html> 标签之前。告知浏览器的解析器，用什么文档类型 规范来解析这个文档。

（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。

2.行内元素有哪些？块级元素有哪些？ 空(void)元素有那些？

（1）CSS规范规定，每个元素都有display属性，确定该元素的类型，每个元素都有默认的display值，比如div默认display属性值为“block”，成为“块级”元素；span默认display属性值为“inline”，是“行内”元素。

（2）行内元素有：a b span img input select strong（强调的语气） 块级元素有：div ul ol li dl dt dd h1 h2 h3 h4…p

（3）知名的空元素： <br> <hr> <img> <input> <link> <meta>

鲜为人知的是： <area> <base> <col> <command> <embed> <keygen> <param> <source> <track> <wbr>

3.CSS的盒子模型？

（1）两种， IE 盒子模型、标准 W3C 盒子模型；IE 的content部分包含了 border 和 pading;

（2）盒模型： 内容(content)、填充(padding)、边界(margin)、 边框(border).

4.link 和@import 的区别是?

（1）、link属于XHTML标签，而@import是CSS提供的;

（2）、页面被加载的时，link会同时被加载，而@import引用的CSS会等到页面被加载完再加载;

（3）、import只在IE5以上才能识别，而link是XHTML标签，无兼容问题;

（4）、link方式的样式的权重 高于@import的权重.

5.CSS 选择符有哪些？哪些属性可以继承？优先级算法如何计算？ CSS3新增伪类有那些？

\*   1.id选择器（ # myid）

2.类选择器（.myclassname）

3.标签选择器（div, h1, p）

4.相邻选择器（h1 + p）

5.子选择器（ul < li）

6.后代选择器（li a）

7.通配符选择器（ \* ）

8.属性选择器（a[rel = "external"]）

9.伪类选择器（a: hover, li: nth - child）

\*   可继承： font-size font-family color, UL LI DL DD DT;

\*   不可继承 ：border padding margin width height ;

\*   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;

优先级为:

!important >id > class > tag

important 比 内联优先级高

CSS3新增伪类举例：

p:first-of-type 选择属于其父元素的首个 <p> 元素的每个 <p> 元素。

p:last-of-type选择属于其父元素的最后 <p> 元素的每个 <p> 元素。

p: only-of-type选择属于其父元素唯一的 <p> 元素的每个 <p> 元素。

p: only-child    选择属于其父元素的唯一子元素的每个 <p> 元素。

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。

:enabled、:disabled 控制表单控件的禁用状态。

:checked，单选框或复选框被选中。

6.如何居中div,如何居中一个浮动元素?

给div设置一个宽度，然后添加margin:0 auto属性

div{

width:200px;

margin:0 auto;

}

居中一个浮动元素

确定容器的宽高 宽500 高 300 的层

设置层的外边距

.div {

Width:500px ; height:300px;//高度可以不设

Margin: -150px 0 0 -250px;

position:relative;相对定位

background-color:pink;//方便看效果

left:50%;

top:50%;

}

7.浏览器的内核分别是什么?经常遇到的浏览器的兼容性有哪些？原因，解决方法是什么，常用hack的技巧 ？

\* IE浏览器的内核Trident、 Mozilla的Gecko、google的WebKit、Opera内核Presto；

\* png24为的图片在iE6浏览器上出现背景，解决方案是做成PNG8.

\* 浏览器默认的margin和padding不同。解决方案是加一个全局的\*{margin:0;padding:0;}来统一。

\* IE6双边距bug:块属性标签float后，又有横行的margin情况下，在ie6显示margin比设置的大。

浮动ie产生的双倍距离 #box{ float:left; width:10px; margin:0 0 0 100px;}

这种情况之下IE会产生20px的距离，解决方案是在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)

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。

css

.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\*/

}

\*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。

\* Chrome 中文界面下默认会将小于 12px 的文本强制按照 12px 显示, 可通过加入 CSS 属性 -webkit-text-size-adjust: none; 解决.

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 {}

8.html5\CSS3有哪些新特性、移除了那些元素？如何处理HTML5新标签的浏览器兼容问题？如何区分 HTML 和 HTML5？

HTML5 现在已经不是 SGML 的子集，主要是关于图像，位置，存储，地理定位等功能的增加。

\* 绘画 canvas 元素

用于媒介回放的 video 和 audio 元素

本地离线存储 localStorage 长期存储数据，浏览器关闭后数据不丢失；

sessionStorage 的数据在浏览器关闭后自动删除

语意化更好的内容元素，比如 article、footer、header、nav、section

表单控件，calendar、date、time、email、url、search

CSS3实现圆角，阴影，对文字加特效，增加了更多的CSS选择器多背景 rgba

新的技术webworker, websockt, Geolocation

移除的元素

纯表现的元素：basefont，big，center，font, s，strike，tt，u；

对可用性产生负面影响的元素：frame，frameset，noframes；

\* 是IE8/IE7/IE6支持通过document.createElement方法产生的标签，

可以利用这一特性让这些浏览器支持HTML5新标签，

浏览器支持新标签后，还需要添加标签默认的样式：

\* 当然最好的方式是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]-->

9.你怎么来实现页面设计图，你认为前端应该如何高质量完成工作? 一个满屏 品 字布局 如何设计?

\* 首先划分成头部、body、脚部；。。。。。

\*

实现效果图是最基本的工作，精确到2px；

与设计师，产品经理的沟通和项目的参与

做好的页面结构，页面重构和用户体验

处理hack，兼容、写出优美的代码格式

针对服务器的优化、拥抱 HTML5。

10.常使用的库有哪些？常用的前端开发工具？开发过什么应用或组件？

\* 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）

\* Sublime Text 、Eclipse、Notepad、Firebug、HttpWatch、Yslow。

\* 城市选择插件，汽车型号选择插件、幻灯片插件。弹出层。（写过开源程序，加载器，js引擎更好）

JavaScript原型，原型链 ? 有什么特点？

\*原型对象也是普通的对象，是对象一个自带隐式的 \_\_proto\_\_ 属性，原型也有可能有自己的原型，如果一个原型对象的原型不为null的话，我们就称之为原型链。

\*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的数据对象有那些属性值？

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。

function clone(proto) {

function Dummy() { }

Dummy.prototype = proto;

Dummy.prototype.constructor = Dummy;

return new Dummy(); //等价于Object.create(Person);

}

function object(old) {

function F() {};

F.prototype = old;

return new F();

}

var newObj = object(oldObject);

12.列出display的值，说明他们的作用。position的值， relative和absolute定位原点是？

1.    block 象块类型元素一样显示。

none 缺省值。向行内元素类型一样显示。

inline-block 象行内元素一样显示，但其内容象块类型元素一样显示。

list-item 象块类型元素一样显示，并添加样式列表标记。

2.

\*absolute

生成绝对定位的元素，相对于 static 定位以外的第一个父元素进行定位。

\*fixed （老IE不支持）

生成绝对定位的元素，相对于浏览器窗口进行定位。

\*relative

生成相对定位的元素，相对于其正常位置进行定位。

\* static默认值。没有定位，元素出现在正常的流中

\*（忽略 top, bottom, left, right z-index 声明）。

\* inherit 规定从父元素继承 position 属性的值。

13.页面重构怎么操作？

编写 CSS、让页面结构更合理化，提升用户体验，实现良好的页面效果和提升性能。

14.语义化的理解？

html语义化就是让页面的内容结构化，便于对浏览器、搜索引擎解析；

在没有样式CCS情况下也以一种文档格式显示，并且是容易阅读的。

搜索引擎的爬虫依赖于标记来确定上下文和各个关键字的权重，利于 SEO。

使阅读源代码的人对网站更容易将网站分块，便于阅读维护理解。

15.HTML5的离线储存？

localStorage    长期存储数据，浏览器关闭后数据不丢失；

sessionStorage数据在浏览器关闭后自动删除。

16.为什么要初始化CSS样式。

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。

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。

\*最简单的初始化方法就是： \* {padding: 0; margin: 0;} （不建议）

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; }

17.(写)描述一段语义的html代码吧。

（HTML5中新增加的很多标签（如：<article>、<nav>、<header>和<footer>等）

就是基于语义化设计原则）

< div id="header">

< h1>标题< /h1>

< h2>专注Web前端技术< /h2>

< /div>

语义 HTML 具有以下特性：

文字包裹在元素中，用以反映内容。例如：

段落包含在 <p> 元素中。

顺序表包含在<ol>元素中。

从其他来源引用的大型文字块包含在<blockquote>元素中。

HTML 元素不能用作语义用途以外的其他目的。例如：

<h1>包含标题，但并非用于放大文本。

<blockquote>包含大段引述，但并非用于文本缩进。

空白段落元素 ( <p></p> ) 并非用于跳行。

文本并不直接包含任何样式信息。例如：

不使用 <font> 或 <center> 等格式标记。

类或 ID 中不引用颜色或位置。

18.absolute的containing block计算方式跟正常流有什么不同？

19.position跟display、margin collapse、overflow、float这些特性相互叠加后会怎么样？

20.对BFC规范的理解？（W3C CSS 2.1 规范中的一个概念,它决定了元素如何对其内容进行定位,以及与其他元素的关 系和相互作用。）

21.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。

22.css定义的权重

以下是权重的规则：标签的权重为1，class的权重为10，id的权重为100，以下例子是演示各种定义的权重值：

/\*权重为1\*/

div{

}

/\*权重为10\*/

.class1{

}

/\*权重为100\*/

#id1{

}

/\*权重为100+1=101\*/

#id1 div{

}

/\*权重为10+1=11\*/

.class1 div{

}

/\*权重为10+10+1=21\*/

.class1 .class2 div{

}

如果权重相同，则最后定义的样式会起作用，但是应该避免这种情况出现

23.eval是做什么的？

它的功能是把对应的字符串解析成JS代码并运行；

避免使用eval，不安全，非常耗性能（2次，一次解析成js语句，一次执行）。

23.写一个通用的事件侦听器函数

`// event(事件)工具集，来源： [https://github.com/markyun](https://github.com/markyun" \t "_blank)

markyun.Event = {

// 页面加载完成后

readyEvent : function(fn) {

if (fn==null) {

fn=document;

}

var oldonload = window.onload;

if (typeof window.onload != 'function') {

window.onload = fn;

} else {

window.onload = function() {

oldonload();

fn();

};

}

},

// 视能力分别使用dom0||dom2||IE方式 来绑定事件

// 参数： 操作的元素,事件名称 ,事件处理程序

addEvent : function(element, type, handler) {

if (element.addEventListener) {

//事件类型、需要执行的函数、是否捕捉

element.addEventListener(type, handler, false);

} else if (element.attachEvent) {

element.attachEvent('on' + type, function() {

handler.call(element);

});

} else {

element['on' + type] = handler;

}

},

// 移除事件

removeEvent : function(element, type, handler) {

if (element.removeEnentListener) {

element.removeEnentListener(type, handler, false);

} else if (element.datachEvent) {

element.detachEvent('on' + type, handler);

} else {

element['on' + type] = null;

}

},

// 阻止事件 (主要是事件冒泡，因为IE不支持事件捕获)

stopPropagation : function(ev) {

if (ev.stopPropagation) {

ev.stopPropagation();

} else {

ev.cancelBubble = true;

}

},

// 取消事件的默认行为

preventDefault : function(event) {

if (event.preventDefault) {

event.preventDefault();

} else {

event.returnValue = false;

}

},

// 获取事件目标

getTarget : function(event) {

return event.target || event.srcElement;

},

// 获取event对象的引用，取到事件的所有信息，确保随时能使用event；

getEvent : function(e) {

var ev = e || window.event;

if (!ev) {

var c = this.getEvent.caller;

while (c) {

ev = c.arguments[0];

if (ev && Event == ev.constructor) {

break;

}

c = c.caller;

}

}

return ev;

}

};

24.99%的网站都需要被重构是那本书上写的？

\* 网站重构：应用web标准进行设计（第2版）

25.什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。

26.Node.js的适用场景

高并发、聊天、实时消息推送

27.WEB应用从服务器主动推送Data到客户端有那些方式？

html5 websoket

WebSocket通过Flash

XHR长时间连接

XHR Multipart Streaming

不可见的Iframe

<script>标签的长时间连接(可跨域)

JavaScript部分

要点：数据类型、面向对象、继承、闭包、插件、作用域、跨域、原型链、模块化、自定义事件、异步装载回调、模板引擎、Nodejs等。

js的几种数据类型：number,string,boolean,object,undefined

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

通常可以做一些小练习来判断TA的水平，js 虽然很灵活，但是具体的代码和实现方式能体现出一个人的全局观，随着代码规模的增长，复杂度增加，如何合理划分模块实现功能和接口的能力比较重要。（下面例题）

["1", "2", "3"].map(parseInt)

[typeof null, null instanceof Object]

[ [3,2,1].reduce(Math.pow), [].reduce(Math.pow)] ]

var val = 'smtg';

console.log('Value is ' + (val === 'smtg') ? 'Something' : 'Nothing');

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.事件、IE与火狐的事件机制有什么区别？ 如何阻止冒泡？

1. 我们在网页中的某个操作（有的操作对应多个事件）。例如：当我们点击一个按钮就会产生一个事件。是可以被 JavaScript 侦测到的行为。

2. 事件处理机制：IE是事件冒泡、火狐是 事件捕获；

3.ev.stopPropagation();

4.什么是闭包（closure），为什么要用？

待完善

执行say667()后,say667()闭包内部变量会存在,而闭包内部函数的内部变量不会存在.使得Javascript的垃圾回收机制 GC不会收回say667()所占用的资源，因为say667()的内部函数的执行需要依赖say667()中的变量。这是对闭包作用的非常直白的描述.

function say667() {

// Local variable that ends up within closure

var num = 666;

var sayAlert = function() { alert(num); }

num++;

return sayAlert;

}

var sayAlert = say667();

sayAlert()//执行结果应该弹出的667

5.如何判断一个对象是否属于某个类？

使用instanceof （待完善）

if(a instanceof Person){

alert('yes');

}

6.new操作符具体干了什么呢?

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、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 。

var obj= {};

obj.\_\_proto\_\_ = Base.prototype;

Base.call(obj);

7.JSON 的了解

JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。它是基于JavaScript的一个子集。数据格式简单, 易于读写, 占用带宽小

{'age':'12', 'name':'back'}

8.js延迟加载的方式有哪些

defer和async、动态创建DOM方式（用得最多）、按需异步载入js

9.ajax 是什么?ajax 的交互模型?同步和异步的区别?如何解决跨域问题?

待完善

1. 通过异步模式，提升了用户体验

2. 优化了浏览器和服务器之间的传输，减少不必要的数据往返，减少了带宽占用

3. Ajax在客户端运行，承担了一部分本来由服务器承担的工作，减少了大用户量下的服务器负载。

2. Ajax的最大的特点是什么。

Ajax可以实现动态不刷新（局部刷新）

readyState属性 状态 有5个可取值： 0=未初始化 ，1=正在加载 2=以加载，3=交互中，4=完成

ajax的缺点

1、ajax不支持浏览器back按钮。

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。

4、破坏了程序的异常机制。

5、不容易调试。

跨域： jsonp、 iframe、window.name、window.postMessage、服务器上设置代理页面

10.模块化怎么做？

立即执行函数,不暴露私有成员

var module1 = (function(){

var \_count = 0;

var m1 = function(){

//...

};

var m2 = function(){

//...

};

return {

m1 : m1,

m2 : m2

};

})();

11.对Node的优点和缺点提出了自己的看法：

\*（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。

12.异步加载的方式

(1) defer，只支持IE

(2) async：

(3) 创建script，插入到DOM中，加载完毕后callBack

documen.write和 innerHTML的区别

document.write只能重绘整个页面

innerHTML可以重绘页面的一部分

13.告诉我答案是多少？

(function(x){

delete x;

alert(x);

})(1+5);

函数参数无法delete删除，delete只能删除通过for in访问的属性。

当然，删除失败也不会报错，所以代码运行会弹出“1”。

14.JS中的call()和apply()方法的区别？

例子中用 add 来替换 sub，add.call(sub,3,1) == add(3,1) ，所以运行结果为：alert(4);

注意：js 中的函数其实是对象，函数名是对 Function 对象的引用。

function add(a,b)

{

alert(a+b);

}

function sub(a,b)

{

alert(a-b);

}

add.call(sub,3,1);

15.Jquery与jQuery UI 有啥区别？

\*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。

\*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

16.jquery 中如何将数组转化为json字符串，然后再转化回来？

jQuery中没有提供这个功能，所以你需要先编写两个jQuery的扩展：

$.fn.stringifyArray = function(array) {

return JSON.stringify(array)

}

$.fn.parseArray = function(array) {

return JSON.parse(array)

}

然后调用：

$("").stringifyArray(array)

17.JavaScript中的作用域与变量声明提升？

其他部分

（HTTP、正则、优化、重构、响应式、移动端、团队协作、SEO、UED、职业生涯）

\*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++) {}

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。

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

一个页面从输入 URL 到页面加载显示完成，这个过程中都发生了什么？（流程说的越详细越好）

你所知道的页面性能优化方法有那些？

除了前端以外还了解什么其它技术么？你最最厉害的技能是什么？

AMD（Modules/Asynchronous-Definition）、CMD（Common Module Definition）规范区别？

18.谈谈你认为怎样做能是项目做的更好？

19.你对前端界面工程师这个职位是怎么样理解的？它的前景会怎么样？

20.加班的看法

加班就像借钱，原则应当是------救急不救穷

21.平时如何管理你的项目，如何设计突发大规模并发架构？

先期团队必须确定好全局样式（globe.css），编码模式(utf-8) 等

编写习惯必须一致（例如都是采用继承式的写法，单样式都写成一行）；

标注样式编写人，各模块都及时标注（标注关键样式调用的地方）；

页面进行标注（例如 页面 模块 开始和结束）；

CSS跟HTML 分文件夹并行存放，命名都得统一（例如style.css）

JS 分文件夹存放 命民以该JS 功能为准英文翻译；

图片采用整合的 images.png png8 格式文件使用 尽量整合在一起使用方便将来的管理

那些操作会造成内存泄漏？

内存泄漏指任何对象在您不再拥有或需要它之后仍然存在。

垃圾回收器定期扫描对象，并计算引用了每个对象的其他对象的数量。如果一个对象的引用数量为 0（没有其他对象引用过该对象），或对该对象的惟一引用是循环的，那么该对象的内存即可回收。

setTimeout 的第一个参数使用字符串而非函数的话，会引发内存泄漏。

闭包、控制台日志、循环（在两个对象彼此引用且彼此保留时，就会产生一个循环）

23.你说你热爱前端，那么应该WEB行业的发展很关注吧？ 说说最近最流行的一些东西吧？

Node.js、Mongodb、npmM、MVVM、MEAN

24.你有了解我们公司吗？说说你的认识？

因为我想去阿里，所以我针对阿里的说

最羡慕就是在双十一购物节，350.19亿元，每分钟支付79万笔。海量数据，居然无一漏单、无一故障。太厉害了。

25.移动端（比如：Android IOS）怎么做好用户体验?

作为一名前端工程师，无论工作年头长短都应该必须掌握的知识点有：

1、DOM结构 —— 两个节点之间可能存在哪些关系以及如何在节点之间任意移动。

2、DOM操作 ——如何添加、移除、移动、复制、创建和查找节点等。

3、事件 —— 如何使用事件，以及IE和标准DOM事件模型之间存在的差别。

4、XMLHttpRequest —— 这是什么、怎样完整地执行一次GET请求、怎样检测错误。

5、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。

6、盒模型 —— 外边距、内边距和边框之间的关系，及IE8以下版本的浏览器中的盒模型

7、块级元素与行内元素 —— 怎么用CSS控制它们、以及如何合理的使用它们

8、浮动元素——怎么使用它们、它们有什么问题以及怎么解决这些问题。

9、HTML与XHTML——二者有什么区别，你觉得应该使用哪一个并说出理由。

10、JSON —— 作用、用途、设计结构。

他们也许不懂交互设计，但是没人比他们懂交互设计的实现，和每一个细节。

他们也许不懂视觉设计，但是没人比他们懂视觉设计如何变为现实。

他们也许不懂后台数据库，但是他们其实才是数据的第一消费者。

他们也许不是产品经理，但是产品的质量几乎都是由他们来决定。

最近也收到许多微博私信的鼓励和更正题目信息，后面会经常更新题目和答案到 [github博客](http://markyun.github.io/" \t "_blank) 。希望前端er达到既能使用也会表达，对理论知识有自己的理解。可根据下面的知识点一个一个去进阶学习，形成自己的职业技能链。

面试有几点需注意：(来源寒冬winter 老师，github:@wintercn)

1. 面试题目： 根据你的等级和职位的变化，入门级到专家级，广度和深度都会有所增加。
2. 题目类型： 理论知识、算法、项目细节、技术视野、开放性题、工作案例。
3. 细节追问： 可以确保问到你开始不懂或面试官开始不懂为止，这样可以大大延展题目的区分度和深度，知道你的实际能力。因为这种知识关联是长时期的学习，临时抱佛脚绝对是记不住的。
4. 回答问题再棒，面试官（可能是你面试职位的直接领导），会考虑我要不要这个人做我的同事？所以态度很重要、除了能做事，还要会做人。（感觉更像是相亲( •̣̣̣̣̣̥́௰•̣̣̣̣̣̥̀ )）
5. 资深的前端开发能把absolute和relative弄混，这样的人不要也罢，因为团队需要的是：你这个人具有可以依靠的才能（靠谱）。

#### 前端开发知识点：

HTML&CSS：

对Web标准的理解、浏览器内核差异、兼容性、hack、CSS基本功：布局、盒子模型、选择器优先级、

HTML5、CSS3、Flexbox

JavaScript：

数据类型、运算、对象、Function、继承、闭包、作用域、原型链、事件、RegExp、JSON、Ajax、

DOM、BOM、内存泄漏、跨域、异步装载、模板引擎、前端MVC、路由、模块化、Canvas、ECMAScript 6、Nodejs

其他：

移动端、响应式、自动化构建、HTTP、离线存储、WEB安全、优化、重构、团队协作、可维护、易用性、SEO、UED、架构、职业生涯、快速学习能力

作为一名前端工程师， **无论工作年头长短都应该掌握的知识点** ：

此条由 王子墨 发表在 [攻城师的实验室](http://lab.yuanwai.wang/" \t "_blank)

1、DOM结构 —— 两个节点之间可能存在哪些关系以及如何在节点之间任意移动。

2、DOM操作 —— 如何添加、移除、移动、复制、创建和查找节点等。

3、事件 —— 如何使用事件，以及IE和标准DOM事件模型之间存在的差别。

4、XMLHttpRequest —— 这是什么、怎样完整地执行一次GET请求、怎样检测错误。

5、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。

6、盒模型 —— 外边距、内边距和边框之间的关系，及IE8以下版本的浏览器中的盒模型

7、块级元素与行内元素 —— 怎么用CSS控制它们、以及如何合理的使用它们

8、浮动元素 —— 怎么使用它们、它们有什么问题以及怎么解决这些问题。

9、HTML与XHTML —— 二者有什么区别，你觉得应该使用哪一个并说出理由。

10、JSON —— 作用、用途、设计结构。

#### 备注：

根据自己需要选择性阅读，面试题是对理论知识的总结，让自己学会应该如何表达。

资料答案不够正确和全面，欢迎欢迎Star和提交issues。

格式不断修改更新中。

### 更新时间: 2015-10-9

## HTML

* Doctype作用？标准模式与兼容模式各有什么区别?
* （1）、<!DOCTYPE>声明位于位于HTML文档中的第一行，处于 <html> 标签之前。告知浏览器的解析器用什么文档标准解析这个文档。DOCTYPE不存在或格式不正确会导致文档以兼容模式呈现。
* （2）、标准模式的排版 和JS运作模式都是以该浏览器支持的最高标准运行。在兼容模式中，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。
* HTML5 为什么只需要写 <!DOCTYPE HTML>？
* HTML5 不基于 SGML，因此不需要对DTD进行引用，但是需要doctype来规范浏览器的行为（让浏览器按照它们应该的方式来运行）；
* 而HTML4.01基于SGML,所以需要对DTD进行引用，才能告知浏览器文档所使用的文档类型。
* 行内元素有哪些？块级元素有哪些？ 空(void)元素有那些？
* 首先：CSS规范规定，每个元素都有display属性，确定该元素的类型，每个元素都有默认的display值，如div的display默认值为“block”，则为“块级”元素；span默认display属性值为“inline”，是“行内”元素。
* （1）行内元素有：a b span img input select strong（强调的语气）
* （2）块级元素有：div ul ol li dl dt dd h1 h2 h3 h4…p
* （3）常见的空元素：
* <br> <hr> <img> <input> <link> <meta>
* 鲜为人知的是：
* <area> <base> <col> <command> <embed> <keygen> <param> <source> <track> <wbr>
* 页面导入样式时，使用link和@import有什么区别？
* （1）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；而@import是CSS提供的，只能用于加载CSS;
* （2）页面被加载的时，link会同时被加载，而@import引用的CSS会等到页面被加载完再加载;
* （3）import是CSS2.1 提出的，只在IE5以上才能被识别，而link是XHTML标签，无兼容问题;
* 介绍一下你对浏览器内核的理解？
* 主要分成两部分：渲染引擎(layout engineer或Rendering Engine)和JS引擎。
* 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引擎则：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。
* 常见的浏览器内核有哪些？
* Trident内核：IE,MaxThon,TT,The World,360,搜狗浏览器等。[又称MSHTML]
* Gecko内核：Netscape6及以上版本，FF,MozillaSuite/SeaMonkey等
* Presto内核：Opera7及以上。 [Opera内核原为：Presto，现为：Blink;]
* Webkit内核：Safari,Chrome等。 [ Chrome的：Blink（WebKit的分支）]

详细文章： [浏览器内核的解析和对比](http://www.cnblogs.com/fullhouse/archive/2011/12/19/2293455.html" \t "_blank)

* html5有哪些新特性、移除了那些元素？如何处理HTML5新标签的浏览器兼容问题？如何区分 HTML 和 HTML5？
* \* 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。
* 绘画 canvas;
* 用于媒介回放的 video 和 audio 元素;
* 本地离线存储 localStorage 长期存储数据，浏览器关闭后数据不丢失;
* sessionStorage 的数据在浏览器关闭后自动删除;
* 语意化更好的内容元素，比如 article、footer、header、nav、section;
* 表单控件，calendar、date、time、email、url、search;
* 新的技术webworker, websockt, Geolocation;
* 移除的元素：
* 纯表现的元素：basefont，big，center，font, s，strike，tt，u;
* 对可用性产生负面影响的元素：frame，frameset，noframes；
* \* 支持HTML5新标签：
* IE8/IE7/IE6支持通过document.createElement方法产生的标签，
* 可以利用这一特性让这些浏览器支持HTML5新标签，
* 浏览器支持新标签后，还需要添加标签默认的样式。
* 当然最好的方式是直接使用成熟的框架、比如html5shim;
* <!--[if lt IE 9]>
* <script> src="http://html5shim.googlecode.com/svn/trunk/html5.js"</script>
* <![endif]-->
* \* 如何区分HTML5： DOCTYPE声明\新增的结构元素\功能元素
* 简述一下你对HTML语义化的理解？
* 用正确的标签做正确的事情。
* html语义化让页面的内容结构化，结构更清晰，便于对浏览器、搜索引擎解析;
* 及时在没有样式CCS情况下也以一种文档格式显示，并且是容易阅读的;
* 搜索引擎的爬虫也依赖于HTML标记来确定上下文和各个关键字的权重，利于SEO;
* 使阅读源代码的人对网站更容易将网站分块，便于阅读维护理解。
* HTML5的离线储存怎么使用，工作原理能不能解释一下？
* 在用户没有与因特网连接时，可以正常访问站点或应用，在用户与因特网连接时，更新用户机器上的缓存文件。
* 原理：HTML5的离线存储是基于一个新建的.appcache文件的缓存机制(不是存储技术)，通过这个文件上的解析清单离线存储资源，这些资源就会像cookie一样被存储了下来。之后当网络在处于离线状态下时，浏览器会通过被离线存储的数据进行页面展示。
* 如何使用：
* 1、页面头部像下面一样加入一个manifest的属性；
* 2、在cache.manifest文件的编写离线存储的资源；
* CACHE MANIFEST
* #v0.11
* CACHE:
* js/app.js
* css/style.css
* NETWORK:
* resourse/logo.png
* FALLBACK:
* / /offline.html
* 3、在离线状态时，操作window.applicationCache进行需求实现。

详细的使用请参考： [有趣的HTML5：离线存储](http://segmentfault.com/a/1190000000732617" \t "_blank)

* 浏览器是怎么对HTML5的离线储存资源进行管理和加载的呢？
* 在线的情况下，浏览器发现html头部有manifest属性，它会请求manifest文件，如果是第一次访问app，那么浏览器就会根据manifest文件的内容下载相应的资源并且进行离线存储。如果已经访问过app并且资源已经离线存储了，那么浏览器就会使用离线的资源加载页面，然后浏览器会对比新的manifest文件与旧的manifest文件，如果文件没有发生改变，就不做任何操作，如果文件改变了，那么就会重新下载文件中的资源并进行离线存储。
* 离线的情况下，浏览器就直接使用离线存储的资源。

详细的使用请参考： [有趣的HTML5：离线存储](http://segmentfault.com/a/1190000000732617" \t "_blank)

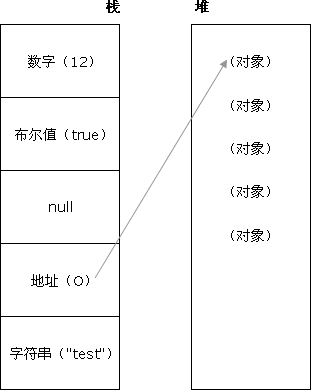
* 请描述一下 cookies，sessionStorage 和 localStorage 的区别？
* cookie是网站为了标示用户身份而储存在用户本地终端（Client Side）上的数据（通常经过加密）。
* cookie数据始终在同源的http请求中携带（即使不需要），记会在浏览器和服务器间来回传递。
* sessionStorage和localStorage不会自动把数据发给服务器，仅在本地保存。
* 存储大小：
* cookie数据大小不能超过4k。
* sessionStorage和localStorage 虽然也有存储大小的限制，但比cookie大得多，可以达到5M或更大。
* 有期时间：
* localStorage 存储持久数据，浏览器关闭后数据不丢失除非主动删除数据；
* sessionStorage 数据在当前浏览器窗口关闭后自动删除。
* cookie 设置的cookie过期时间之前一直有效，即使窗口或浏览器关闭
* iframe有那些缺点？
* \*iframe会阻塞主页面的Onload事件；
* \*搜索引擎的检索程序无法解读这种页面，不利于SEO;
* \*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，这样可以绕开以上两个问题。
* Label的作用是什么？是怎么用的？
* label标签来定义表单控制间的关系,当用户选择该标签时，浏览器会自动将焦点转到和标签相关的表单控件上。
* <label for="Name">Number:</label>
* <input type=“text“name="Name" id="Name"/>
* <label>Date:<input type="text" name="B"/></label>
* HTML5的form如何关闭自动完成功能？
* 给不想要提示的 form 或下某个input 设置为 autocomplete=off。
* 如何实现浏览器内多个标签页之间的通信? (阿里)
* 调用localstorge、cookies等本地存储方式
* webSocket如何兼容低浏览器？(阿里)
* Adobe Flash Socket 、
* ActiveX HTMLFile (IE) 、
* 基于 multipart 编码发送 XHR 、
* 基于长轮询的 XHR
* 页面可见性（Page Visibility）API 可以有哪些用途？
* 在页面被切换到其他后台进程的时候，自动暂停音乐或视频的播放；
* 如何在页面上实现一个圆形的可点击区域？
* 1、map+area或者svg
* 2、border-radius
* 3、纯js实现 需要求一个点在不在圆上简单算法、获取鼠标坐标等等
* 实现不使用 border 画出1px高的线，在不同浏览器的标准模式与怪异模式下都能保持一致的效果。
* <div style="height:1px;overflow:hidden;background:#ccc"></div>
* 网页验证码是干嘛的，是为了解决什么安全问题。
* 区分用户是计算机还是人的公共全自动程序。可以防止：恶意破解密码、刷票、论坛灌水；
* 有效防止黑客对某一个特定注册用户用特定程序暴力破解方式进行不断的登陆尝试；

## CSS

* 介绍一下CSS的盒子模型？
* （1）有两种， IE 盒子模型、标准 W3C 盒子模型；IE的content部分包含了 border 和 pading;
* （2）盒模型： 内容(content)、填充(padding)、边界(margin)、 边框(border).
* CSS选择符有哪些？哪些属性可以继承？
* \* 1.id选择器（ # myid）
* 2.类选择器（.myclassname）
* 3.标签选择器（div, h1, p）
* 4.相邻选择器（h1 + p）
* 5.子选择器（ul > li）
* 6.后代选择器（li a）
* 7.通配符选择器（ \* ）
* 8.属性选择器（a[rel = "external"]）
* 9.伪类选择器（a: hover, li: nth - child）
* \* 可继承的样式： font-size font-family color, UL LI DL DD DT;
* \* 不可继承的样式：border padding margin width height ;
* CSS优先级算法如何计算？ \* 优先级就近原则，同权重情况下样式定义最近者为准;
* \* 载入样式以最后载入的定位为准;
* 优先级为:
* !important > id > class > tag
* important 比 内联优先级高
* CSS3新增伪类有那些？
* CSS3新增伪类举例：
* p:first-of-type 选择属于其父元素的首个 <p> 元素的每个 <p> 元素。
* p:last-of-type 选择属于其父元素的最后 <p> 元素的每个 <p> 元素。
* p:only-of-type 选择属于其父元素唯一的 <p> 元素的每个 <p> 元素。
* p:only-child 选择属于其父元素的唯一子元素的每个 <p> 元素。
* 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。
* :enabled :disabled 控制表单控件的禁用状态。
* :checked 单选框或复选框被选中。
* 如何居中div？如何居中一个浮动元素？如何让绝对定位的div居中？
  + 给div设置一个宽度，然后添加margin:0 auto属性
  + div{
  + width:200px;
  + margin:0 auto;
  + }
  + 居中一个浮动元素
  + 确定容器的宽高 宽500 高 300 的层
  + 设置层的外边距
  + .div {
  + width:500px ; height:300px;//高度可以不设
  + margin: -150px 0 0 -250px;
  + position:relative; //相对定位
  + background-color:pink; //方便看效果
  + left:50%;
  + top:50%;
  + }
  + 让绝对定位的div居中
  + position: absolute;
  + width: 1200px;
  + background: none;
  + margin: 0 auto;
  + top: 0;
  + left: 0;
  + bottom: 0;
  + right: 0;
* display有哪些值？说明他们的作用。
* block 象块类型元素一样显示。
* none 缺省值。象行内元素类型一样显示。
* inline-block 象行内元素一样显示，但其内容象块类型元素一样显示。
* list-item 象块类型元素一样显示，并添加样式列表标记。
* position的值relative和absolute定位原点是？
* absolute
* 生成绝对定位的元素，相对于 static 定位以外的第一个父元素进行定位。
* fixed （老IE不支持）
* 生成绝对定位的元素，相对于浏览器窗口进行定位。
* relative
* 生成相对定位的元素，相对于其正常位置进行定位。
* static
* 默认值。没有定位，元素出现在正常的流中
* （忽略 top, bottom, left, right z-index 声明）。
* inherit
* 规定从父元素继承 position 属性的值。
* CSS3有哪些新特性？
* CSS3实现圆角（border-radius:8px），
* 阴影（box-shadow:10px），
* 文字特效（text-shadow、），
* 线性渐变（gradient），
* 旋转（transform）
* transform:rotate(9deg) scale(0.85,0.90) translate(0px,-30px) skew(-9deg,0deg);//旋转,缩放,定位,倾斜
* 增加了更多的CSS选择器
* 多背景 rgba
* 请解释一下CSS3的Flexbox（弹性盒布局模型）,以及适用场景？
* 用纯CSS创建一个三角形的原理是什么？
* 把上、左、右三条边隐藏掉（颜色设为 transparent）
* #demo {
* width: 0;
* height: 0;
* border-width: 20px;
* border-style: solid;
* border-color: transparent transparent red transparent;
* }
* 一个满屏 品 字布局 如何设计?
* 简单的方式：
* 上面的div宽100%，
* 下面的两个div分别宽50%，
* 然后用float或者inline使其不换行即可
* 常见兼容性问题？
* \* png24位的图片在iE6浏览器上出现背景，解决方案是做成PNG8.
* \* 浏览器默认的margin和padding不同。解决方案是加一个全局的\*{margin:0;padding:0;}来统一。
* \* IE6双边距bug:块属性标签float后，又有横行的margin情况下，在ie6显示margin比设置的大。
* 浮动ie产生的双倍距离 #box{ float:left; width:10px; margin:0 0 0 100px;}
* 这种情况之下IE会产生20px的距离，解决方案是在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)
* 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。
* css
* .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\*/
* }
* \* 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。
* \* 解决方法：（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。
* \* Chrome 中文界面下默认会将小于 12px 的文本强制按照 12px 显示,
* 可通过加入 CSS 属性 -webkit-text-size-adjust: none; 解决。
* 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 {}
* li与li之间有看不见的空白间隔是什么原因引起的？有什么解决办法？
* 经常遇到的浏览器的兼容性有哪些？原因，解决方法是什么，常用hack的技巧 ？
* 为什么要初始化CSS样式。
* - 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。
* - 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。
* 最简单的初始化方法： \* {padding: 0; margin: 0;} （强烈不建议）
* 淘宝的样式初始化代码：
* 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; }
* absolute的containing block(容器块)计算方式跟正常流有什么不同？
* 无论属于哪种，都要先找到其祖先元素中最近的 position 值不为 static 的元素，然后再判断：
* 1、若此元素为 inline 元素，则 containing block 为能够包含这个元素生成的第一个和最后一个 inline box 的 padding box (除 margin, border 外的区域) 的最小矩形；
* 2、否则,则由这个祖先元素的 padding box 构成。
* 如果都找不到，则为 initial containing block。
* 补充：
* 1. static(默认的)/relative：简单说就是它的父元素的内容框（即去掉padding的部分）
* 2. absolute: 向上找最近的定位为absolute/relative的元素
* 3. fixed: 它的containing block一律为根元素(html/body)，根元素也是initial containing block
* CSS里的visibility属性有个collapse属性值是干嘛用的？在不同浏览器下以后什么区别？
* position跟display、margin collapse、overflow、float这些特性相互叠加后会怎么样？
* 对BFC规范(块级格式化上下文：block formatting context)的理解？
* （W3C CSS 2.1 规范中的一个概念,它是一个独立容器，决定了元素如何对其内容进行定位,以及与其他元素的关系和相互作用。）
* 一个页面是由很多个 Box 组成的,元素的类型和 display 属性,决定了这个 Box 的类型。
* 不同类型的 Box,会参与不同的 Formatting Context（决定如何渲染文档的容器）,因此Box内的元素会以不同的方式渲染,也就是说BFC内部的元素和外部的元素不会互相影响。
* css定义的权重
* 以下是权重的规则：标签的权重为1，class的权重为10，id的权重为100，以下例子是演示各种定义的权重值：
* /\*权重为1\*/
* div{
* }
* /\*权重为10\*/
* .class1{
* }
* /\*权重为100\*/
* #id1{
* }
* /\*权重为100+1=101\*/
* #id1 div{
* }
* /\*权重为10+1=11\*/
* .class1 div{
* }
* /\*权重为10+10+1=21\*/
* .class1 .class2 div{
* }
* 如果权重相同，则最后定义的样式会起作用，但是应该避免这种情况出现
* 请解释一下为什么会出现浮动和什么时候需要清除浮动？清除浮动的方式
* 移动端的布局用过媒体查询吗？
* 使用 CSS 预处理器吗？喜欢那个？
* SASS (SASS、LESS没有本质区别，只因为团队前端都是用的SASS)
* CSS优化、提高性能的方法有哪些？
* 浏览器是怎样解析CSS选择器的？
* 在网页中的应该使用奇数还是偶数的字体？为什么呢？
* margin和padding分别适合什么场景使用？
* 抽离样式模块怎么写，说出思路，有无实践经验？[阿里航旅的面试题]
* 元素竖向的百分比设定是相对于容器的高度吗？
* 全屏滚动的原理是什么？用到了CSS的那些属性？
* 什么是响应式设计？响应式设计的基本原理是什么？如何兼容低版本的IE？
* 视差滚动效果，如何给每页做不同的动画？（回到顶部，向下滑动要再次出现，和只出现一次分别怎么做？）
* ::before 和 :after中双冒号和单冒号 有什么区别？解释一下这2个伪元素的作用。
* 如何修改chrome记住密码后自动填充表单的黄色背景 ？
* 你对line-height是如何理解的？
* 设置元素浮动后，该元素的display值是多少？（自动变成display:block）
* 怎么让Chrome支持小于12px 的文字？
* 让页面里的字体变清晰，变细用CSS怎么做？（-webkit-font-smoothing: antialiased;）
* font-style属性可以让它赋值为“oblique” oblique是什么意思？
* position:fixed;在android下无效怎么处理？
* 如果需要手动写动画，你认为最小时间间隔是多久，为什么？（阿里）
* 多数显示器默认频率是60Hz，即1秒刷新60次，所以理论上最小间隔为1/60＊1000ms ＝ 16.7ms
* display:inline-block 什么时候会显示间隙？(携程)
* 移除空格、使用margin负值、使用font-size:0、letter-spacing、word-spacing
* overflow: scroll时不能平滑滚动的问题怎么处理？
* 有一个高度自适应的div，里面有两个div，一个高度100px，希望另一个填满剩下的高度。
* png、jpg、gif 这些图片格式解释一下，分别什么时候用。有没有了解过webp？
* 什么是Cookie 隔离？（或者说：请求资源的时候不要让它带cookie怎么做）
* 如果静态文件都放在主域名下，那静态文件请求的时候都带有的cookie的数据提交给server的，非常浪费流量，
* 所以不如隔离开。
* 因为cookie有域的限制，因此不能跨域提交请求，故使用非主要域名的时候，请求头中就不会带有cookie数据，
* 这样可以降低请求头的大小，降低请求时间，从而达到降低整体请求延时的目的。
* 同时这种方式不会将cookie传入Web Server，也减少了Web Server对cookie的处理分析环节，
* 提高了webserver的http请求的解析速度。

## JavaScript

* 介绍js的基本数据类型。
* Undefined、Null、Boolean、Number、String
* 介绍js有哪些内置对象？
* Object 是 JavaScript 中所有对象的父对象
* 数据封装类对象：Object、Array、Boolean、Number 和 String
* 其他对象：Function、Arguments、Math、Date、RegExp、Error
* 说几条写JavaScript的基本规范？
* 1.不要在同一行声明多个变量。
* 2.请使用 ===/!==来比较true/false或者数值
* 3.使用对象字面量替代new Array这种形式
* 4.不要使用全局函数。
* 5.Switch语句必须带有default分支
* 6.函数不应该有时候有返回值，有时候没有返回值。
* 7.For循环必须使用大括号
* 8.If语句必须使用大括号
* 9.for-in循环中的变量 应该使用var关键字明确限定作用域，从而避免作用域污染。
* JavaScript原型，原型链 ? 有什么特点？
* 每个对象都会在其内部初始化一个属性，就是prototype(原型)，当我们访问一个对象的属性时，
* 如果这个对象内部不存在这个属性，那么他就会去prototype里找这个属性，这个prototype又会有自己的prototype，
* 于是就这样一直找下去，也就是我们平时所说的原型链的概念。
* 关系：instance.constructor.prototype = instance.\_\_proto\_\_
* 特点：
* JavaScript对象是通过引用来传递的，我们创建的每个新对象实体中并没有一份属于自己的原型副本。当我们修改原型时，与之相关的对象也会继承这一改变。
* 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，如此递推下去，一直检索到 Object 内建对象。
* function Func(){}
* Func.prototype.name = "Sean";
* Func.prototype.getInfo = function() {
* return this.name;
* }
* var person = new Func();//现在可以参考var person = Object.create(oldObject);
* console.log(person.getInfo());//它拥有了Func的属性和方法
* //"Sean"
* console.log(Func.prototype);
* // Func { name="Sean", getInfo=function()}
* JavaScript有几种类型的值？，你能画一下他们的内存图吗？
* 栈：原始数据类型（Undefined，Null，Boolean，Number、String）
* 堆：引用数据类型（对象、数组和函数）
* 两种类型的区别是：存储位置不同；
* 原始数据类型直接存储在栈(stack)中的简单数据段，占据空间小、大小固定，属于被频繁使用数据，所以放入栈中存储；
* 引用数据类型存储在堆(heap)中的对象,占据空间大、大小不固定,如果存储在栈中，将会影响程序运行的性能；引用数据类型在栈中存储了指针，该指针指向堆中该实体的起始地址。当解释器寻找引用值时，会首先检索其
* 在栈中的地址，取得地址后从堆中获得实体



* Javascript如何实现继承？
* 1、构造继承
* 2、原型继承
* 3、实例继承
* 4、拷贝继承
* 原型prototype机制或apply和call方法去实现较简单，建议使用构造函数与原型混合方式。
* function Parent(){
* this.name = 'wang';
* }
* function Child(){
* this.age = 28;
* }
* Child.prototype = new Parent();//继承了Parent，通过原型
* var demo = new Child();
* alert(demo.age);
* alert(demo.name);//得到被继承的属性
* }
* javascript创建对象的几种方式？
* javascript创建对象简单的说,无非就是使用内置对象或各种自定义对象，当然还可以用JSON；但写法有很多种，也能混合使用。
* 1、对象字面量的方式
* person={firstname:"Mark",lastname:"Yun",age:25,eyecolor:"black"};
* 2、用function来模拟无参的构造函数
* function Person(){}
* var person=new Person();//定义一个function，如果使用new"实例化",该function可以看作是一个Class
* person.name="Mark";
* person.age="25";
* person.work=function(){
* alert(person.name+" hello...");
* }
* person.work();
* 3、用function来模拟参构造函数来实现（用this关键字定义构造的上下文属性）
* function Pet(name,age,hobby){
* this.name=name;//this作用域：当前对象
* this.age=age;
* this.hobby=hobby;
* this.eat=function(){
* alert("我叫"+this.name+",我喜欢"+this.hobby+",是个程序员");
* }
* }
* var maidou =new Pet("麦兜",25,"coding");//实例化、创建对象
* maidou.eat();//调用eat方法
* 4、用工厂方式来创建（内置对象）
* var wcDog =new Object();
* wcDog.name="旺财";
* wcDog.age=3;
* wcDog.work=function(){
* alert("我是"+wcDog.name+",汪汪汪......");
* }
* wcDog.work();
* 5、用原型方式来创建
* function Dog(){
* }
* Dog.prototype.name="旺财";
* Dog.prototype.eat=function(){
* alert(this.name+"是个吃货");
* }
* var wangcai =new Dog();
* wangcai.eat();
* 5、用混合方式来创建
* function Car(name,price){
* this.name=name;
* this.price=price;
* }
* Car.prototype.sell=function(){
* alert("我是"+this.name+"，我现在卖"+this.price+"万元");
* }
* var camry =new Car("凯美瑞",27);
* camry.sell();
* Javascript作用链域?
* 全局函数无法查看局部函数的内部细节，但局部函数可以查看其上层的函数细节，直至全局细节。
* 当需要从局部函数查找某一属性或方法时，如果当前作用域没有找到，就会上溯到上层作用域查找，
* 直至全局函数，这种组织形式就是作用域链。
* 谈谈This对象的理解。
* eval是做什么的？
* 它的功能是把对应的字符串解析成JS代码并运行；
* 应该避免使用eval，不安全，非常耗性能（2次，一次解析成js语句，一次执行）。
* 什么是window对象? 什么是document对象?
* null，undefined 的区别？
* null 表示一个对象被定义了，值为“空值”；
* undefined 表示不存在这个值。
* typeof undefined
* //"undefined"
* undefined :是一个表示"无"的原始值或者说表示"缺少值"，就是此处应该有一个值，但是还没有定义。当尝试读取时会返回 undefined；
* 例如变量被声明了，但没有赋值时，就等于undefined
* typeof null
* //"object"
* null : 是一个对象(空对象, 没有任何属性和方法)；
* 例如作为函数的参数，表示该函数的参数不是对象；
* 注意：
* 在验证null时，一定要使用　=== ，因为 == 无法分别 null 和　undefined
* 再来一个例子：
* null
* Q：有张三这个人么？
* A：有！
* Q：张三有房子么？
* A：没有！
* undefined
* Q：有张三这个人么？
* A：没有！
* 写一个通用的事件侦听器函数。
* // event(事件)工具集，来源：github.com/markyun
* markyun.Event = {
* // 页面加载完成后
* readyEvent : function(fn) {
* if (fn==null) {
* fn=document;
* }
* var oldonload = window.onload;
* if (typeof window.onload != 'function') {
* window.onload = fn;
* } else {
* window.onload = function() {
* oldonload();
* fn();
* };
* }
* },
* // 视能力分别使用dom0||dom2||IE方式 来绑定事件
* // 参数： 操作的元素,事件名称 ,事件处理程序
* addEvent : function(element, type, handler) {
* if (element.addEventListener) {
* //事件类型、需要执行的函数、是否捕捉
* element.addEventListener(type, handler, false);
* } else if (element.attachEvent) {
* element.attachEvent('on' + type, function() {
* handler.call(element);
* });
* } else {
* element['on' + type] = handler;
* }
* },
* // 移除事件
* removeEvent : function(element, type, handler) {
* if (element.removeEventListener) {
* element.removeEventListener(type, handler, false);
* } else if (element.datachEvent) {
* element.detachEvent('on' + type, handler);
* } else {
* element['on' + type] = null;
* }
* },
* // 阻止事件 (主要是事件冒泡，因为IE不支持事件捕获)
* stopPropagation : function(ev) {
* if (ev.stopPropagation) {
* ev.stopPropagation();
* } else {
* ev.cancelBubble = true;
* }
* },
* // 取消事件的默认行为
* preventDefault : function(event) {
* if (event.preventDefault) {
* event.preventDefault();
* } else {
* event.returnValue = false;
* }
* },
* // 获取事件目标
* getTarget : function(event) {
* return event.target || event.srcElement;
* },
* // 获取event对象的引用，取到事件的所有信息，确保随时能使用event；
* getEvent : function(e) {
* var ev = e || window.event;
* if (!ev) {
* var c = this.getEvent.caller;
* while (c) {
* ev = c.arguments[0];
* if (ev && Event == ev.constructor) {
* break;
* }
* c = c.caller;
* }
* }
* return ev;
* }
* };
* ["1", "2", "3"].map(parseInt) 答案是多少？
* [1, NaN, NaN] 因为 parseInt 需要两个参数 (val, radix)，
* 其中 radix 表示解析时用的基数。
* map 传了 3 个 (element, index, array)，对应的 radix 不合法导致解析失败。
* 事件是？IE与火狐的事件机制有什么区别？ 如何阻止冒泡？
* 1. 我们在网页中的某个操作（有的操作对应多个事件）。例如：当我们点击一个按钮就会产生一个事件。是可以被 JavaScript 侦测到的行为。
* 2. 事件处理机制：IE是事件冒泡、Firefox同时支持两种事件模型，也就是：捕获型事件和冒泡型事件；
* 3. ev.stopPropagation();（旧ie的方法 ev.cancelBubble = true;）
* 什么是闭包（closure），为什么要用它？
* 闭包是指有权访问另一个函数作用域中变量的函数，创建闭包的最常见的方式就是在一个函数内创建另一个函数，通过另一个函数访问这个函数的局部变量,利用闭包可以突破作用链域，将函数内部的变量和方法传递到外部。
* 闭包的特性：
* 1.函数内再嵌套函数
* 2.内部函数可以引用外层的参数和变量
* 3.参数和变量不会被垃圾回收机制回收
* //li节点的onclick事件都能正确的弹出当前被点击的li索引
* <ul id="testUL">
* <li> index = 0</li>
* <li> index = 1</li>
* <li> index = 2</li>
* <li> index = 3</li>
* </ul>
* <script type="text/javascript">
* var nodes = document.getElementsByTagName("li");
* for(i = 0;i<nodes.length;i+= 1){
* nodes[i].onclick = function(){
* console.log(i+1);//不用闭包的话，值每次都是4
* }(i);
* }
* </script>
* 执行say667()后,say667()闭包内部变量会存在,而闭包内部函数的内部变量不会存在
* 使得Javascript的垃圾回收机制GC不会收回say667()所占用的资源
* 因为say667()的内部函数的执行需要依赖say667()中的变量
* 这是对闭包作用的非常直白的描述
* function say667() {
* // Local variable that ends up within closure
* var num = 666;
* var sayAlert = function() {
* alert(num);
* }
* num++;
* return sayAlert;
* }
* var sayAlert = say667();
* sayAlert()//执行结果应该弹出的667
* javascript 代码中的"use strict";是什么意思 ? 使用它区别是什么？
* use strict是一种ECMAscript 5 添加的（严格）运行模式,这种模式使得 Javascript 在更严格的条件下运行,
* 使JS编码更加规范化的模式,消除Javascript语法的一些不合理、不严谨之处，减少一些怪异行为。
* 默认支持的糟糕特性都会被禁用，比如不能用with，也不能在意外的情况下给全局变量赋值;
* 全局变量的显示声明,函数必须声明在顶层，不允许在非函数代码块内声明函数,arguments.callee也不允许使用；
* 消除代码运行的一些不安全之处，保证代码运行的安全,限制函数中的arguments修改，严格模式下的eval函数的行为和非严格模式的也不相同;
* 提高编译器效率，增加运行速度；
* 为未来新版本的Javascript标准化做铺垫。
* 如何判断一个对象是否属于某个类？
* 使用instanceof （待完善）
* if(a instanceof Person){
* alert('yes');
* }
* new操作符具体干了什么呢?
* 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、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 。
* var obj = {};
* obj.\_\_proto\_\_ = Base.prototype;
* Base.call(obj);
* 用原生JavaScript的实现过什么功能吗？
* Javascript中，有一个函数，执行时对象查找时，永远不会去查找原型，这个函数是？
* hasOwnProperty
* javaScript中hasOwnProperty函数方法是返回一个布尔值，指出一个对象是否具有指定名称的属性。此方法无法检查该对象的原型链中是否具有该属性；该属性必须是对象本身的一个成员。
* 使用方法：
* object.hasOwnProperty(proName)
* 其中参数object是必选项。一个对象的实例。
* proName是必选项。一个属性名称的字符串值。
* 如果 object 具有指定名称的属性，那么JavaScript中hasOwnProperty函数方法返回 true，反之则返回 false。
* JSON 的了解？
* JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。
* 它是基于JavaScript的一个子集。数据格式简单, 易于读写, 占用带宽小
* 如：{"age":"12", "name":"back"}
* [].forEach.call($$("\*"),function(a){a.style.outline="1px solid #"+(~~(Math.random()\*(1<<24))).toString(16)}) 能解释一下这段代码的意思吗？
* js延迟加载的方式有哪些？
* defer和async、动态创建DOM方式（用得最多）、按需异步载入js
* Ajax 是什么? 如何创建一个Ajax？
* ajax的全称：Asynchronous Javascript And XML。
* 异步传输+js+xml。
* 所谓异步，在这里简单地解释就是：向服务器发送请求的时候，我们不必等待结果，而是可以同时做其他的事情，等到有了结果它自己会根据设定进行后续操作，与此同时，页面是不会发生整页刷新的，提高了用户体验。
* (1)创建XMLHttpRequest对象,也就是创建一个异步调用对象
* (2)创建一个新的HTTP请求,并指定该HTTP请求的方法、URL及验证信息
* (3)设置响应HTTP请求状态变化的函数
* (4)发送HTTP请求
* (5)获取异步调用返回的数据
* (6)使用JavaScript和DOM实现局部刷新
* 同步和异步的区别?

同步：浏览器访问服务器请求，用户看得到页面刷新，重新发请求,等请求完，页面刷新，新内容出现，用户看到新内容,j进行下一步操作。

异步：浏览器访问服务器请求，用户正常操作，浏览器后端进行请求。等请求完，页面不刷新，新内容也会出现，用户看到新内容。

（待完善）

* 如何解决跨域问题?
* jsonp、 iframe、window.name、window.postMessage、服务器上设置代理页面
* 页面编码和被请求的资源编码如果不一致如何处理？
* 模块化开发怎么做？

[立即执行函数](http://benalman.com/news/2010/11/immediately-invoked-function-expression/" \t "_blank) ,不暴露私有成员

var module1 = (function(){

　　　　var \_count = 0;

　　　　var m1 = function(){

　　　　　　//...

　　　　};

　　　　var m2 = function(){

　　　　　　//...

　　　　};

　　　　return {

　　　　　　m1 : m1,

　　　　　　m2 : m2

　　　　};

　　})();

（待完善）

* AMD（Modules/Asynchronous-Definition）、CMD（Common Module Definition）规范区别？

AMD 规范在这里： [https://github.com/amdjs/amdjs-api/wiki/AMD](https://github.com/amdjs/amdjs-api/wiki/AMD" \t "_blank)

CMD 规范在这里： [https://github.com/seajs/seajs/issues/242](https://github.com/seajs/seajs/issues/242" \t "_blank)

Asynchronous Module Definition，异步模块定义，所有的模块将被异步加载，模块加载不影响后面语句运行。所有依赖某些模块的语句均放置在回调函数中。

区别：

1. 对于依赖的模块，AMD 是提前执行，CMD 是延迟执行。不过 RequireJS 从 2.0 开始，也改成可以延迟执行（根据写法不同，处理方式不同）。CMD 推崇 as lazy as possible.

2. CMD 推崇依赖就近，AMD 推崇依赖前置。看代码：

// CMD

define(function(require, exports, module) {

var a = require('./a')

a.doSomething()

// 此处略去 100 行

var b = require('./b') // 依赖可以就近书写

b.doSomething()

// ...

})

// AMD 默认推荐

define(['./a', './b'], function(a, b) { // 依赖必须一开始就写好

a.doSomething()

// 此处略去 100 行

b.doSomething()

// ...

})

* requireJS的核心原理是什么？（如何动态加载的？如何避免多次加载的？如何 缓存的？）
* 谈一谈你对ECMAScript6的了解？
* ECMAScript6 怎么写class么，为什么会出现class这种东西?
* 异步加载JS的方式有哪些？
* (1) defer，只支持IE
* (2) async：
* (3) 创建script，插入到DOM中，加载完毕后callBack
* documen.write和 innerHTML的区别
* document.write只能重绘整个页面
* innerHTML可以重绘页面的一部分
* DOM操作——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
* appendChild()
* removeChild()
* replaceChild()
* insertBefore() //在已有的子节点前插入一个新的子节点
* （3）查找
* getElementsByTagName() //通过标签名称
* getElementsByName() //通过元素的Name属性的值(IE容错能力较强，会得到一个数组，其中包括id等于name值的)
* getElementById() //通过元素Id，唯一性
* .call() 和 .apply() 的区别？
* 例子中用 add 来替换 sub，add.call(sub,3,1) == add(3,1) ，所以运行结果为：alert(4);
* 注意：js 中的函数其实是对象，函数名是对 Function 对象的引用。
* function add(a,b)
* {
* alert(a+b);
* }
* function sub(a,b)
* {
* alert(a-b);
* }
* add.call(sub,3,1);
* 数组和对象有哪些原生方法，列举一下？
* JS 怎么实现一个类。怎么实例化这个类
* JavaScript中的作用域与变量声明提升？
* 如何编写高性能的Javascript？
* 那些操作会造成内存泄漏？
* JQuery的源码看过吗？能不能简单概况一下它的实现原理？
* jQuery.fn的init方法返回的this指的是什么对象？为什么要返回this？
* jquery中如何将数组转化为json字符串，然后再转化回来？
* jQuery 的属性拷贝(extend)的实现原理是什么，如何实现深拷贝？
* jquery.extend 与 jquery.fn.extend的区别？
* jQuery 的队列是如何实现的？队列可以用在哪些地方？
* 谈一下Jquery中的bind(),live(),delegate(),on()的区别？
* JQuery一个对象可以同时绑定多个事件，这是如何实现的？
* 是否知道自定义事件。jQuery里的fire函数是什么意思，什么时候用？
* jQuery 是通过哪个方法和 Sizzle 选择器结合的？（jQuery.fn.find()进入Sizzle）
* 针对 jQuery性能的优化方法？
* Jquery与jQuery UI 有啥区别？
* \*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。
* \*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
* JQuery的源码看过吗？能不能简单说一下它的实现原理？
* jquery 中如何将数组转化为json字符串，然后再转化回来？

jQuery中没有提供这个功能，所以你需要先编写两个jQuery的扩展：

$.fn.stringifyArray = function(array) {

return JSON.stringify(array)

}

$.fn.parseArray = function(array) {

return JSON.parse(array)

}

然后调用：

$("").stringifyArray(array)

* jQuery和Zepto的区别？各自的使用场景？
* 针对 jQuery 的优化方法？
* \*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++) {}
* Zepto的点透问题如何解决？
* jQueryUI如何自定义组件?
* 需求：实现一个页面操作不会整页刷新的网站，并且能在浏览器前进、后退时正确响应。给出你的技术实现方案？
* 如何判断当前脚本运行在浏览器还是node环境中？（阿里）
* 通过判断Global对象是否为window，如果不为window，当前脚本没有运行在浏览器中
* 移动端最小触控区域是多大？
* jQuery 的 slideUp动画 ，如果目标元素是被外部事件驱动, 当鼠标快速地连续触发外部元素事件, 动画会滞后的反复执行，该如何处理呢?
* 把 Script 标签 放在页面的最底部的body封闭之前 和封闭之后有什么区别？浏览器会如何解析它们？
* 移动端的点击事件的有延迟，时间是多久，为什么会有？ 怎么解决这个延时？（click 有 300ms 延迟,为了实现safari的双击事件的设计，浏览器要知道你是不是要双击操作。）
* 知道各种JS框架(Angular, Backbone, Ember, React, Meteor, Knockout...)么? 能讲出他们各自的优点和缺点么?
* Underscore 对哪些 JS 原生对象进行了扩展以及提供了哪些好用的函数方法？
* 解释JavaScript中的作用域与变量声明提升？
* 那些操作会造成内存泄漏？
* 内存泄漏指任何对象在您不再拥有或需要它之后仍然存在。
* 垃圾回收器定期扫描对象，并计算引用了每个对象的其他对象的数量。如果一个对象的引用数量为 0（没有其他对象引用过该对象），或对该对象的惟一引用是循环的，那么该对象的内存即可回收。
* setTimeout 的第一个参数使用字符串而非函数的话，会引发内存泄漏。
* 闭包、控制台日志、循环（在两个对象彼此引用且彼此保留时，就会产生一个循环）
* JQuery一个对象可以同时绑定多个事件，这是如何实现的？
* Node.js的适用场景？
* (如果会用node)知道route, middleware, cluster, nodemon, pm2, server-side rendering么?
* 解释一下 Backbone 的 MVC 实现方式？
* 什么是“前端路由”?什么时候适合使用“前端路由”? “前端路由”有哪些优点和缺点?
* 知道什么是webkit么? 知道怎么用浏览器的各种工具来调试和debug代码么?
* 如何测试前端代码么? 知道BDD, TDD, Unit Test么? 知道怎么测试你的前端工程么(mocha, sinon, jasmin, qUnit..)?
* 前端templating(Mustache, underscore, handlebars)是干嘛的, 怎么用?
* 简述一下 Handlebars 的基本用法？
* 简述一下 Handlerbars 的对模板的基本处理流程， 如何编译的？如何缓存的？
* 用js实现千位分隔符?(来源： [前端农民工](http://div.io/topic/744" \t "_blank) ，提示：正则+replace)
* function commafy(num) {
* num = num + '';
* var reg = /(-?d+)(d{3})/;
* if(reg.test(num)){
* num = num.replace(reg, '$1,$2');
* }
* return num;
* }
* 检测浏览器版本版本有哪些方式？
* 功能检测、userAgent特征检测
* 比如：navigator.userAgent
* //"Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_10\_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/41.0.2272.101 Safari/537.36"

## 其他问题

* 原来公司工作流程是怎么样的，如何与其他人协作的？如何夸部门合作的？
* 你遇到过比较难的技术问题是？你是如何解决的？
* 设计模式 知道什么是singleton, factory, strategy, decrator么?
* 常使用的库有哪些？常用的前端开发工具？开发过什么应用或组件？
* 页面重构怎么操作？
* 网站重构：在不改变外部行为的前提下，简化结构、添加可读性，而在网站前端保持一致的行为。
* 也就是说是在不改变UI的情况下，对网站进行优化，在扩展的同时保持一致的UI。
* 对于传统的网站来说重构通常是：
* 表格(table)布局改为DIV+CSS
* 使网站前端兼容于现代浏览器(针对于不合规范的CSS、如对IE6有效的)
* 对于移动平台的优化
* 针对于SEO进行优化
* 深层次的网站重构应该考虑的方面
* 减少代码间的耦合
* 让代码保持弹性
* 严格按规范编写代码
* 设计可扩展的API
* 代替旧有的框架、语言(如VB)
* 增强用户体验
* 通常来说对于速度的优化也包含在重构中
* 压缩JS、CSS、image等前端资源(通常是由服务器来解决)
* 程序的性能优化(如数据读写)
* 采用CDN来加速资源加载
* 对于JS DOM的优化
* HTTP服务器的文件缓存
* 列举IE与其他浏览器不一样的特性？
* 1、事件不同之处：
* 触发事件的元素被认为是目标（target）。而在 IE 中，目标包含在 event 对象的 srcElement 属性；
* 获取字符代码、如果按键代表一个字符（shift、ctrl、alt除外），IE 的 keyCode 会返回字符代码（Unicode），DOM 中按键的代码和字符是分离的，要获取字符代码，需要使用 charCode 属性；
* 阻止某个事件的默认行为，IE 中阻止某个事件的默认行为，必须将 returnValue 属性设置为 false，Mozilla 中，需要调用 preventDefault() 方法；
* 停止事件冒泡，IE 中阻止事件进一步冒泡，需要设置 cancelBubble 为 true，Mozzilla 中，需要调用 stopPropagation()；
* 99%的网站都需要被重构是那本书上写的？
* 网站重构：应用web标准进行设计（第2版）
* 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，则代码会针对旧版本的IE进行降级处理了,使之在旧式浏览器上以某种形式降级体验却不至于完全不能用。
* 如：border-shadow
* 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版本浏览器才支持的功能,向页面增加不影响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。
* 如：默认使用flash上传，但如果浏览器支持 HTML5 的文件上传功能，则使用HTML5实现更好的体验；
* 是否了解公钥加密和私钥加密。
* 一般情况下是指私钥用于对数据进行签名，公钥用于对签名进行验证;
* HTTP网站在浏览器端用公钥加密敏感数据，然后在服务器端再用私钥解密。
* WEB应用从服务器主动推送Data到客户端有那些方式？
* html5提供的Websocket
* 不可见的iframe
* WebSocket通过Flash
* XHR长时间连接
* XHR Multipart Streaming
* <script>标签的长时间连接(可跨域)
* 对Node的优点和缺点提出了自己的看法？
* \*（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。
* 你有用过哪些前端性能优化的方法？
* （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。
* 对普通的网站有一个统一的思路，就是尽量向前端优化、减少数据库操作、减少磁盘IO。向前端优化指的是，在不影响功能和体验的情况下，能在浏览器执行的不要在服务端执行，能在缓存服务器上直接返回的不要到应用服务器，程序能直接取得的结果不要到外部取得，本机内能取得的数据不要到远程取，内存能取到的不要到磁盘取，缓存中有的不要去数据库查询。减少数据库操作指减少更新次数、缓存结果减少查询次数、将数据库执行的操作尽可能的让你的程序完成（例如join查询），减少磁盘IO指尽量不使用文件系统作为缓存、减少读写文件次数等。程序优化永远要优化慢的部分，换语言是无法“优化”的。
* http状态码有那些？分别代表是什么意思？
* 简单版
* [
* 100 Continue 继续，一般在发送post请求时，已发送了http header之后服务端将返回此信息，表示确认，之后发送具体参数信息
* 200 OK 正常返回信息
* 201 Created 请求成功并且服务器创建了新的资源
* 202 Accepted 服务器已接受请求，但尚未处理
* 301 Moved Permanently 请求的网页已永久移动到新位置。
* 302 Found 临时性重定向。
* 303 See Other 临时性重定向，且总是使用 GET 请求新的 URI。
* 304 Not Modified 自从上次请求后，请求的网页未修改过。
* 400 Bad Request 服务器无法理解请求的格式，客户端不应当尝试再次使用相同的内容发起请求。
* 401 Unauthorized 请求未授权。
* 403 Forbidden 禁止访问。
* 404 Not Found 找不到如何与 URI 相匹配的资源。
* 500 Internal Server Error 最常见的服务器端错误。
* 503 Service Unavailable 服务器端暂时无法处理请求（可能是过载或维护）。
* ]
* 完整版
* 1\*\*(信息类)：表示接收到请求并且继续处理
* 100——客户必须继续发出请求
* 101——客户要求服务器根据请求转换HTTP协议版本
* 2\*\*(响应成功)：表示动作被成功接收、理解和接受
* 200——表明该请求被成功地完成，所请求的资源发送回客户端
* 201——提示知道新文件的URL
* 202——接受和处理、但处理未完成
* 203——返回信息不确定或不完整
* 204——请求收到，但返回信息为空
* 205——服务器完成了请求，用户代理必须复位当前已经浏览过的文件
* 206——服务器已经完成了部分用户的GET请求
* 3\*\*(重定向类)：为了完成指定的动作，必须接受进一步处理
* 300——请求的资源可在多处得到
* 301——本网页被永久性转移到另一个URL
* 302——请求的网页被转移到一个新的地址，但客户访问仍继续通过原始URL地址，重定向，新的URL会在response中的Location中返回，浏览器将会使用新的URL发出新的Request。
* 303——建议客户访问其他URL或访问方式
* 304——自从上次请求后，请求的网页未修改过，服务器返回此响应时，不会返回网页内容，代表上次的文档已经被缓存了，还可以继续使用
* 305——请求的资源必须从服务器指定的地址得到
* 306——前一版本HTTP中使用的代码，现行版本中不再使用
* 307——申明请求的资源临时性删除
* 4\*\*(客户端错误类)：请求包含错误语法或不能正确执行
* 400——客户端请求有语法错误，不能被服务器所理解
* 401——请求未经授权，这个状态代码必须和WWW-Authenticate报头域一起使用
* HTTP 401.1 - 未授权：登录失败
* HTTP 401.2 - 未授权：服务器配置问题导致登录失败
* HTTP 401.3 - ACL 禁止访问资源
* HTTP 401.4 - 未授权：授权被筛选器拒绝
* HTTP 401.5 - 未授权：ISAPI 或 CGI 授权失败
* 402——保留有效ChargeTo头响应
* 403——禁止访问，服务器收到请求，但是拒绝提供服务
* HTTP 403.1 禁止访问：禁止可执行访问
* HTTP 403.2 - 禁止访问：禁止读访问
* HTTP 403.3 - 禁止访问：禁止写访问
* HTTP 403.4 - 禁止访问：要求 SSL
* HTTP 403.5 - 禁止访问：要求 SSL 128
* HTTP 403.6 - 禁止访问：IP 地址被拒绝
* HTTP 403.7 - 禁止访问：要求客户证书
* HTTP 403.8 - 禁止访问：禁止站点访问
* HTTP 403.9 - 禁止访问：连接的用户过多
* HTTP 403.10 - 禁止访问：配置无效
* HTTP 403.11 - 禁止访问：密码更改
* HTTP 403.12 - 禁止访问：映射器拒绝访问
* HTTP 403.13 - 禁止访问：客户证书已被吊销
* HTTP 403.15 - 禁止访问：客户访问许可过多
* HTTP 403.16 - 禁止访问：客户证书不可信或者无效
* HTTP 403.17 - 禁止访问：客户证书已经到期或者尚未生效
* 404——一个404错误表明可连接服务器，但服务器无法取得所请求的网页，请求资源不存在。eg：输入了错误的URL
* 405——用户在Request-Line字段定义的方法不允许
* 406——根据用户发送的Accept拖，请求资源不可访问
* 407——类似401，用户必须首先在代理服务器上得到授权
* 408——客户端没有在用户指定的饿时间内完成请求
* 409——对当前资源状态，请求不能完成
* 410——服务器上不再有此资源且无进一步的参考地址
* 411——服务器拒绝用户定义的Content-Length属性请求
* 412——一个或多个请求头字段在当前请求中错误
* 413——请求的资源大于服务器允许的大小
* 414——请求的资源URL长于服务器允许的长度
* 415——请求资源不支持请求项目格式
* 416——请求中包含Range请求头字段，在当前请求资源范围内没有range指示值，请求也不包含If-Range请求头字段
* 417——服务器不满足请求Expect头字段指定的期望值，如果是代理服务器，可能是下一级服务器不能满足请求长。
* 5\*\*(服务端错误类)：服务器不能正确执行一个正确的请求
* HTTP 500 - 服务器遇到错误，无法完成请求
* HTTP 500.100 - 内部服务器错误 - ASP 错误
* HTTP 500-11 服务器关闭
* HTTP 500-12 应用程序重新启动
* HTTP 500-13 - 服务器太忙
* HTTP 500-14 - 应用程序无效
* HTTP 500-15 - 不允许请求 global.asa
* Error 501 - 未实现
* HTTP 502 - 网关错误
* HTTP 503：由于超载或停机维护，服务器目前无法使用，一段时间后可能恢复正常
* 一个页面从输入 URL 到页面加载显示完成，这个过程中都发生了什么？（流程说的越详细越好）
* 注：这题胜在区分度高，知识点覆盖广，再不懂的人，也能答出几句，
* 而高手可以根据自己擅长的领域自由发挥，从URL规范、HTTP协议、DNS、CDN、数据库查询、
* 到浏览器流式解析、CSS规则构建、layout、paint、onload/domready、JS执行、JS API绑定等等；
* 详细版：
* 1、浏览器会开启一个线程来处理这个请求，对 URL 分析判断如果是 http 协议就按照 Web 方式来处理;
* 2、调用浏览器内核中的对应方法，比如 WebView 中的 loadUrl 方法;
* 3、通过DNS解析获取网址的IP地址，设置 UA 等信息发出第二个GET请求;
* 4、进行HTTP协议会话，客户端发送报头(请求报头);
* 5、进入到web服务器上的 Web Server，如 Apache、Tomcat、Node.JS 等服务器;
* 6、进入部署好的后端应用，如 PHP、Java、JavaScript、Python 等，找到对应的请求处理;
* 7、处理结束回馈报头，此处如果浏览器访问过，缓存上有对应资源，会与服务器最后修改时间对比，一致则返回304;
* 8、浏览器开始下载html文档(响应报头，状态码200)，同时使用缓存;
* 9、文档树建立，根据标记请求所需指定MIME类型的文件（比如css、js）,同时设置了cookie;
* 10、页面开始渲染DOM，JS根据DOM API操作DOM,执行事件绑定等，页面显示完成。
* 简洁版：
* 浏览器根据请求的URL交给DNS域名解析，找到真实IP，向服务器发起请求；
* 服务器交给后台处理完成后返回数据，浏览器接收文件（HTML、JS、CSS、图象等）；
* 浏览器对加载到的资源（HTML、JS、CSS等）进行语法解析，建立相应的内部数据结构（如HTML的DOM）；
* 载入解析到的资源文件，渲染页面，完成。