华清远见

2019年面试题整理

移动互联网

目录

[华清远见 1](#_Toc21869)

[2018年面试题整理 1](#_Toc29023)

[移动互联网 1](#_Toc24874)

[HTML 12](#_Toc17495)

[Doctype 作用？标准模式与兼容模式各有什么区别? 12](#_Toc19158)

[HTML5 为什么只需要写 <!DOCTYPE HTML>？ 12](#_Toc4153)

[行内元素有哪些？块级元素有哪些？ 空(void)元素有那些？ 12](#_Toc15008)

[页面导入样式时，使用 link 和@import 有什么区别？ 12](#_Toc31782)

[介绍一下你对浏览器内核的理解？ 13](#_Toc9232)

[html5 有哪些新特性、移除了那些元素？ 13](#_Toc6862)

[如何处理 HTML5 新标签的浏览器兼容问题？ 13](#_Toc3923)

[如何区分 HTML 和 HTML5？ 14](#_Toc31480)

[简述一下你对 HTML 语义化的理解？ 14](#_Toc8729)

[HTML5 的离线储存怎么使用，工作原理能不能解释一下？ 14](#_Toc23930)

[浏览器是怎么对 HTML5 的离线储存资源进行管理和加载的呢？ 15](#_Toc3131)

[请描述一下 cookies，sessionStorage 和 localStorage 的区别？ 15](#_Toc17332)

[iframe 有那些缺点？ 15](#_Toc24264)

[Label 的作用是什么？是怎么用的？ 16](#_Toc25346)

[HTML5 的 form 如何关闭自动完成功能？ 16](#_Toc18465)

[如何实现浏览器内多个标签页之间的通信? (阿里) 16](#_Toc11472)

[webSocket 如何兼容低浏览器？(阿里) 16](#_Toc17996)

[页面可见性（Page Visibility API） 可以有哪些用途？ 16](#_Toc22575)

[如何在页面上实现一个圆形的可点击区域？ 16](#_Toc10417)

[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。 17](#_Toc279)

[网页验证码是干嘛的，是为了解决什么安全问题。 17](#_Toc17621)

[title 与 h1 的区别、b 与 strong 的区别、i 与 em 的区别？ 17](#_Toc21180)

[b, i, u, s, pre 17](#_Toc7632)

[html 中 title 属性和 alt 属性的区别？ 17](#_Toc7593)

[浏览器内核 18](#_Toc31889)

[1，Trident 18](#_Toc32627)

[2，Gecko 18](#_Toc10032)

[3，Presto 18](#_Toc12611)

[4，Webkit 18](#_Toc6106)

[5，Blink 18](#_Toc31634)

[6，edge 18](#_Toc22561)

[什么叫优雅降级和渐进增强？ 18](#_Toc192)

[CSS 19](#_Toc2990)

[display: none; 与 visibility: hidden; 的区别 19](#_Toc23923)

[css hack 原理及常用 hack 19](#_Toc19882)

[link 与 @import 的区别 21](#_Toc21728)

[CSS 有哪些继承属性 21](#_Toc24159)

[display,float,position 的关系 21](#_Toc12396)

[外边距折叠(collapsing margins) 22](#_Toc19160)

[介绍一下标准的 CSS 的盒子模型？低版本 IE 的盒子模型有什么不同的？ 22](#_Toc30025)

[CSS 选择符有哪些？ 22](#_Toc18059)

[CSS3 新增伪类有那些？ 23](#_Toc26613)

[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？ 23](#_Toc28502)

[如何竖直居中一个元素 25](#_Toc21707)

[display 有哪些值？说明他们的作用 25](#_Toc13903)

[position 有哪些值 relative 和 absolute 定位原点是？ 26](#_Toc3592)

[CSS3 有哪些新特性？ 26](#_Toc15422)

[用纯 CSS 创建一个三角形的原理是什么？ 26](#_Toc21224)

[一个满屏品字布局如何设计? 27](#_Toc9912)

[经常遇到的浏览器的兼容性有哪些？原因，解决方法，常用 hack 的技巧 ？ 27](#_Toc12494)

[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？ 27](#_Toc23343)

[display:inline-block 间隙问题怎么解决？(携程) 27](#_Toc13972)

[display:inline-block 什么时候会显示间隙？ 28](#_Toc14202)

[css 定义的权重 28](#_Toc21367)

[CSS 优先级算法如何计算？ 29](#_Toc23456)

[谈谈浮动和清除浮动 30](#_Toc17977)

[box-sizing 常用的属性有哪些？分别有什么作用？ 30](#_Toc30437)

[请列举几种隐藏元素的方法 31](#_Toc26217)

[rgba() 和 opacity 的透明效果有什么不同？ 31](#_Toc23155)

[css 属性 content 有什么作用？ 31](#_Toc8446)

[请解释一下 CSS3 的 Flexbox（弹性盒布局模型）以及适用场景？ 31](#_Toc26132)

[请写出多种等高布局 31](#_Toc2718)

[圣杯布局的实现原理？ 31](#_Toc18381)

[什么是双飞翼布局？实现原理？ 32](#_Toc15689)

[在 CSS 样式中常使用 px、em 在表现上有什么区别？ 33](#_Toc3432)

[为什么要初始化 CSS 样式？ 33](#_Toc9287)

[reset.css 和 Normalize.css 理解 33](#_Toc16166)

[什么是 FOUC(Flash of Unstyled Content)？ 如何来避免 FOUC？ 35](#_Toc12418)

[介绍使用过的 CSS 预处理器？ 35](#_Toc13621)

[CSS 优化、提高性能的方法有哪些？ 35](#_Toc182)

[浏览器是怎样解析 CSS 选择器的？ 35](#_Toc9628)

[在网页中的应该使用奇数还是偶数的字体？ 35](#_Toc9243)

[margin 和 padding 分别适合什么场景使用？ 36](#_Toc5818)

[抽离样式模块怎么写，说出思路？ 36](#_Toc4986)

[元素竖向的百分比设定是相对于容器的高度吗？ 36](#_Toc653)

[全屏滚动的原理是什么？ 用到了 CSS 的那些属性？ 36](#_Toc7117)

[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？ 36](#_Toc6487)

[什么是视差滚动效果，如何给每页做不同的动画？ 37](#_Toc3788)

[a 标签上四个伪类的执行顺序是怎么样的？ 37](#_Toc18020)

[伪元素和伪类的区别和作用？ 37](#_Toc28964)

[::before 和 :after 中双冒号和单冒号有什么区别？ 38](#_Toc28666)

[如何修改 Chrome 记住密码后自动填充表单的黄色背景？ 38](#_Toc23338)

[网站图片文件，如何点击下载？而非点击预览？ 38](#_Toc29619)

[iOS safari 如何阻止“橡皮筋效果”？ 38](#_Toc10301)

[你对 line-height 是如何理解的？ 39](#_Toc10971)

[line-height 三种赋值方式有何区别？（带单位、纯数字、百分比） 39](#_Toc12865)

[设置元素浮动后，该元素的 display 值会如何变化？ 39](#_Toc24195)

[怎么让 Chrome 支持小于 12px 的文字？ 39](#_Toc463)

[让页面里的字体变清晰，变细用 CSS 怎么做？（IOS 手机浏览器字体齿轮设置） 39](#_Toc4457)

[如果需要手动写动画，你认为最小时间间隔是多久？ 40](#_Toc24848)

[overflow: scroll 时不能平滑滚动的问题怎么处理？ 40](#_Toc30184)

[CSS 中类 class 和 id 的区别 40](#_Toc28264)

[如何优化网页的打印样式 40](#_Toc22659)

[请问为何要使用 transform 而非 absolute positioning，或反之的理由？为什么？ 40](#_Toc30027)

[请解释 CSS sprites，以及你要如何在页面或网站中实现它 41](#_Toc19416)

[你熟悉 SVG 样式的书写吗？ 41](#_Toc18169)

[如果设计中使用了非标准的字体，你该如何去实现？ 41](#_Toc2284)

[JavaScript基础 42](#_Toc23037)

[JavaScript 的组成 42](#_Toc20284)

[JS 的基本数据类型和引用数据类型 42](#_Toc28309)

[检测浏览器版本版本有哪些方式？ 42](#_Toc10745)

[介绍 JS 有哪些内置对象？ 42](#_Toc10098)

[说几条写 JavaScript 的基本规范？ 42](#_Toc27720)

[如何编写高性能的 JavaScript？ 43](#_Toc16758)

[offsetWidth/offsetHeight,clientWidth/clientHeight 与 scrollWidth/scrollHeight 的区别 43](#_Toc1165)

[描述浏览器的渲染过程，DOM 树和渲染树的区别？ 43](#_Toc10836)

[重绘和回流（重排）的区别和关系？ 44](#_Toc1800)

[如何最小化重绘(repaint)和回流(reflow)？ 44](#_Toc30017)

[script 的位置是否会影响首屏显示时间？ 44](#_Toc4679)

[解释 JavaScript 中的作用域与变量声明提升？ 44](#_Toc1084)

[介绍 JavaScript 的原型，原型链？有什么特点？ 45](#_Toc20106)

[JavaScript 有几种类型的值？你能画一下他们的内存图吗 45](#_Toc27898)

[JavaScript 如何实现一个类，怎么实例化这个类？ 45](#_Toc5771)

[Javascript 如何实现继承？ 47](#_Toc21606)

[js 继承方式及其优缺点 48](#_Toc20379)

[javascript 创建对象的几种方式？ 48](#_Toc21086)

[eval 是做什么的？ 49](#_Toc29922)

[事件的三个阶段 50](#_Toc13601)

[介绍事件“捕获”和“冒泡”执行顺序和事件的执行次数？ 50](#_Toc17521)

[在一个 DOM 上同时绑定两个点击事件：一个用捕获，一个用冒泡。事件会执行几次，先执行冒泡还是捕获？ 50](#_Toc4101)

[事件的代理/委托 50](#_Toc17327)

[IE 与火狐的事件机制有什么区别？ 如何阻止冒泡？ 51](#_Toc7348)

[IE 的事件处理和 W3C 的事件处理有哪些区别？(必考) 51](#_Toc20488)

[W3C 事件的 target 与 currentTarget 的区别？ 51](#_Toc27536)

[什么是函数节流？介绍一下应用场景和原理？ 52](#_Toc26208)

[区分什么是“客户区坐标”、“页面坐标”、“屏幕坐标”？ 52](#_Toc999)

[如何获得一个 DOM 元素的绝对位置？ 52](#_Toc24281)

[分析 ['1', '2', '3'].map(parseInt) 答案是多少？（常考） 52](#_Toc23330)

[new 操作符具体干了什么？ 53](#_Toc2013)

[用原生 JavaScript 的实现过什么功能吗？ 53](#_Toc534)

[什么是闭包（closure），为什么要用它？ 53](#_Toc8864)

[javascript 代码中的"use strict";是什么意思 ? 使用它区别是什么？ 54](#_Toc8725)

[js 延迟加载的方式有哪些？ 54](#_Toc5225)

[defer 和 async 54](#_Toc28547)

[同步和异步的区别? 54](#_Toc31586)

[documen.write 和 innerHTML 的区别 54](#_Toc9708)

[说说你对闭包的理解 54](#_Toc2761)

[请解释一下 JavaScript 的同源策略 55](#_Toc496)

[为什么要有同源限制？ 55](#_Toc29103)

[说说严格模式的限制 56](#_Toc3977)

[编写一个方法 求一个字符串的字节长度 56](#_Toc26793)

[假设：一个英文字符占用一个字节，一个中文字符占用两个字节 56](#_Toc5998)

[请解释什么是事件代理 56](#_Toc9765)

[attribute 和 property 的区别是什么？ 57](#_Toc3233)

[页面编码和被请求的资源编码如果不一致如何处理？ 57](#_Toc13883)

[把 <script> 放在 </body> 之前和之后有什么区别？浏览器会如何解析它们？ 57](#_Toc9072)

[把 <script> 放在 </head> 中会有什么问题？ 57](#_Toc23093)

[JavaScript 中，调用函数有哪几种方式？ 57](#_Toc17772)

[简单实现 Function.bind 函数？ 58](#_Toc29985)

[列举一下 JavaScript 数组和对象有哪些原生方法 58](#_Toc29219)

[Array.slice() 与 Array.splice() 的区别？ 59](#_Toc30250)

[在 javascript 中，1 与 Number(1)有什么区别 [易混淆] 59](#_Toc23201)

[console.log(!!(new Boolean(false))输出什么 [易混淆] 59](#_Toc30349)

[浏览器中的 Event Loop 60](#_Toc13826)

[正则表达式 60](#_Toc12390)

[JavaScript高级 61](#_Toc32673)

[请指出document load和document ready的区别？ 61](#_Toc23719)

[DOM 元素 e 的 e.getAttribute(propName)和 e.propName 有什么区别和联系 61](#_Toc5029)

[什么是 Window 对象? 什么是 Document 对象? 62](#_Toc14831)

[介绍 DOM 的发展 62](#_Toc1266)

[介绍 DOM0，DOM2，DOM3 事件处理方式区别 62](#_Toc19247)

[JavaScript 实现异步编程的方法？ 62](#_Toc8153)

[DOM 操作——怎样添加、移除、移动、复制、创建和查找节点? 63](#_Toc5163)

[用过哪些设计模式？ 63](#_Toc26352)

[1. 工厂模式： 63](#_Toc2915)

[2. 构造函数模式 63](#_Toc1014)

[那些操作会造成内存泄漏？ 64](#_Toc29672)

[Javascript 垃圾回收方法 64](#_Toc13503)

[为什么 JS 是单线程,而不是多线程 [常考] 64](#_Toc10019)

[JavaScript 对象生命周期的理解？ 65](#_Toc15469)

[如何派发事件(dispatchEvent)？（如何进行事件广播？） 65](#_Toc21919)

[Javascript 作用链域? 65](#_Toc25476)

[数据结构 65](#_Toc16211)

[什么是数据结构？ 65](#_Toc9633)

[8种常用数据结构 66](#_Toc9978)

[撤回，即Ctrl+Z，你知道它是怎么实现的吗？ 66](#_Toc25045)

[jQuery 66](#_Toc8577)

[Ajax 是什么? 如何创建一个 Ajax？ 66](#_Toc19192)

[Ajax的优缺点及工作原理？ 66](#_Toc10643)

[jQuery 库中的 $() 是什么？ 67](#_Toc32622)

[如何找到所有 HTML select 标签的选中项？ 67](#_Toc24775)

[$(this) 和 this 关键字在 jQuery 中有何不同？ 67](#_Toc11313)

[jquery怎么移除标签onclick属性？ 67](#_Toc29439)

[jquery中addClass,removeClass,toggleClass的使用。 67](#_Toc32037)

[JQuery有几种选择器? 68](#_Toc30195)

[jQuery中的Delegate()函数有什么作用？ 68](#_Toc27896)

[$(document).ready()方法和window.onload有什么区别？ 68](#_Toc11787)

[如何用jQuery禁用浏览器的前进后退按钮？ 68](#_Toc22816)

[jquery中$.get()提交和$.post()提交有区别吗？ 69](#_Toc10982)

[写出一个简单的$.ajax()的请求方式？ 69](#_Toc8887)

[jQuery的事件委托方法bind 、live、delegate、on之间有什么区别？ 69](#_Toc325)

[BootStrap 70](#_Toc11077)

[1.为什么使用bootstrap? 70](#_Toc21818)

[2.为了让 Bootstrap 开发的网站对移动设备友好，确保适当的绘制和触屏缩放，需要在网页的 head 之中添加 viewport meta 标签，width 属性控制设备的宽度。 70](#_Toc13488)

[3.对于各类尺寸的设备，Bootstrap设置的class前缀分别是什么？ 71](#_Toc16063)

[4.Bootstrap如何设置响应式表格？ 71](#_Toc9714)

[5、使用Bootstrap创建垂直表单的基本步骤？ 71](#_Toc9256)

[6、使用Bootstrap创建水平表单的基本步骤？ 71](#_Toc585)

[7、使用Bootstrap如何创建表单控件的帮助文本？ 71](#_Toc16655)

[8、使用Bootstrap激活或禁用按钮要如何操作？ 71](#_Toc3822)

[9、Bootstrap有哪些关于<img>的class？ 72](#_Toc1448)

[10、Bootstrap中有关元素浮动及清除浮动的class？ 72](#_Toc17318)

[11、除了屏幕阅读器外，其他设备上隐藏元素的class？ 72](#_Toc6554)

[12、Bootstrap如何制作下拉菜单？ 72](#_Toc17568)

[13、Bootstrap如何制作按钮组？以及水平按钮组和垂直按钮组的优先级？ 72](#_Toc26329)

[14、Bootstrap如何设置按钮的下拉菜单？ 72](#_Toc22653)

[15、Bootstrap中的输入框组如何制作？ 73](#_Toc29715)

[16、Bootstrap中的导航都有哪些？ 73](#_Toc9087)

[17、Bootstrap中设置分页的class？ 73](#_Toc21412)

[18、Bootstrap中显示标签的class？ 73](#_Toc31014)

[19、Bootstrap中如何制作徽章？ 73](#_Toc21310)

[NodeJS 73](#_Toc24522)

[为什么用Nodejs，它有哪些缺点？ 73](#_Toc14748)

[sessionStorage 、localStorage 和 cookie 之间的区别 74](#_Toc32242)

[Web Storage与Cookie相比存在的优势： 74](#_Toc2810)

[web 开发中会话跟踪的方法有哪些 74](#_Toc10105)

[如何删除一个 cookie 75](#_Toc21033)

[什么是错误优先的回调函数？ 75](#_Toc12478)

[如何避免回调地狱 75](#_Toc7038)

[如何用Node监听80端口 75](#_Toc8307)

[什么是事件循环 76](#_Toc30694)

[使用NPM有哪些好处？ 77](#_Toc23752)

[什么是Stub？举个使用场景 77](#_Toc20963)

[什么是测试金字塔？ 77](#_Toc5889)

[React 78](#_Toc27463)

[React 中 keys 的作用是什么？ 78](#_Toc4289)

[调用 setState 之后发生了什么？ 79](#_Toc28803)

[react 生命周期函数 79](#_Toc19071)

[shouldComponentUpdate 是做什么的，（react 性能优化是哪个周期函数？） 79](#_Toc18211)

[为什么虚拟 dom 会提高性能?(必考) 79](#_Toc27872)

[react diff 原理（常考，大厂必考） 80](#_Toc13922)

[React 中 refs 的作用是什么？ 80](#_Toc16545)

[如果你创建了类似于下面的 Twitter 元素，那么它相关的类定义是啥样子的？ 81](#_Toc26820)

[展示组件(Presentational component)和容器组件(Container component)之间有何不同 82](#_Toc27175)

[类组件(Class component)和函数式组件(Functional component)之间有何不同 82](#_Toc19781)

[(组件的)状态(state)和属性(props)之间有何不同 82](#_Toc12293)

[何为受控组件(controlled component) 82](#_Toc23980)

[何为高阶组件(higher order component) 83](#_Toc9335)

[为什么建议传递给 setState 的参数是一个 callback 而不是一个对象 83](#_Toc11400)

[除了在构造函数中绑定 this，还有其它方式吗 83](#_Toc23533)

[(在构造函数中)调用 super(props) 的目的是什么 83](#_Toc8227)

[应该在 React 组件的何处发起 Ajax 请求 83](#_Toc8381)

[描述事件在 React 中的处理方式。 83](#_Toc10730)

[createElement 和 cloneElement 有什么区别？ 84](#_Toc21864)

[React 中有三种构建组件的方式 84](#_Toc1551)

[react 组件的划分业务组件技术组件？ 84](#_Toc546)

[简述 flux 思想 84](#_Toc11653)

[React 项目用过什么脚手架（本题是开放性题目） 85](#_Toc12151)

[了解 redux 么，说一下 redux 吧 85](#_Toc13437)

[redux 有什么缺点 85](#_Toc32583)

[Vue 85](#_Toc25092)

[1、active-class是哪个组件的属性？嵌套路由怎么定义？ 85](#_Toc15983)

[2、怎么定义vue-router的动态路由？怎么获取传过来的动态参数？  85](#_Toc29054)

[3、vue-router有哪几种导航钩子？   86](#_Toc15567)

[4、scss是什么？在vue.cli中的安装使用步骤是？有哪几大特性？ 86](#_Toc13584)

[5、mint-ui是什么？怎么使用？说出至少三个组件使用方法？ 86](#_Toc26270)

[6、v-model是什么？怎么使用？ vue中标签怎么绑定事件？ 86](#_Toc18484)

[7、axios是什么？怎么使用？描述使用它实现登录功能的流程？ 86](#_Toc12575)

[8、axios+tp5进阶中，调用axios.post(‘api/user’)是进行的什么操作？axios.put(‘api/user/8′)呢？ 87](#_Toc15371)

[9、什么是RESTful API？怎么使用? 87](#_Toc15921)

[10、vuex是什么？怎么使用？哪种功能场景使用它？ 87](#_Toc13100)

[11、mvvm框架是什么？它和其它框架（jquery）的区别是什么？哪些场景适合？ 87](#_Toc30049)

[12、自定义指令（v-check、v-focus）的方法有哪些？它有哪些钩子函数？还有哪些钩子函数参数？ 87](#_Toc20737)

[13、说出至少4种vue当中的指令和它的用法？ 87](#_Toc1621)

[14、vue-router是什么？它有哪些组件？ 88](#_Toc16622)

[15、导航钩子有哪些？它们有哪些参数？ 88](#_Toc10191)

[16、Vue的双向数据绑定原理是什么？ 88](#_Toc31707)

[17、请详细说下你对vue生命周期的理解？ 88](#_Toc16285)

[18、请说下封装 vue 组件的过程？ 89](#_Toc26323)

[19、你是怎么认识vuex的？ 89](#_Toc17198)

[20、vue-loader是什么？使用它的用途有哪些？ 89](#_Toc31702)

[21、请说出vue.cli项目中src目录每个文件夹和文件的用法？ 89](#_Toc23972)

[22、vue.cli中怎样使用自定义的组件？有遇到过哪些问题吗？ 89](#_Toc30231)

[23、聊聊你对Vue.js的template编译的理解？ 90](#_Toc3992)

[综合 90](#_Toc1332)

[1、规避javascript多人开发函数重名问题 90](#_Toc24051)

[2、请说出三种减低页面加载时间的方法 90](#_Toc26415)

[3、你所了解到的Web攻击技术 90](#_Toc4223)

[4、web前端开发，如何提高页面性能优化？ 91](#_Toc1842)

[5、前端开发中，如何优化图像？图像格式的区别？ 91](#_Toc17957)

[6、浏览器是如何渲染页面的？ 92](#_Toc28509)

# HTML

## Doctype 作用？标准模式与兼容模式各有什么区别?

* ​声明位于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。​
* 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（标准通用标记语言），因此不需要对 DTD（文档类型定义）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）；<https://blog.csdn.net/weixin_39072332/article/details/79304622>
* 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？

定义：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。

* 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
* 空元素：
  + 常见: br hr img input link meta
  + 不常见: area base col command embed keygen param source track wbr

不同浏览器（版本）、HTML4（5）、CSS2 等实际略有差异 参考: <http://stackoverflow.com/questions/6867254/browsers-default-css-for-html-elements>

## 页面导入样式时，使用 link 和@import 有什么区别？

* 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;
* 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;
* 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;
* link 支持使用 js 控制 DOM 去改变样式，而@import 不支持;

## 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)

## html5 有哪些新特性、移除了那些元素？

* 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, websocket, 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；

## 如何处理 HTML5 新标签的浏览器兼容问题？

* 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。
  + 当然也可以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 HTML 和 HTML5？

* 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;
* 即使在没有样式 CSS 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  
#v1.0  
​  
CACHE:  
js/app.js  
css/style.css  
​  
NETWORK:  
assets/logo.png  
​  
FALLBACK:  
/html5/ /404.html

1. 在离线状态时，操作 window.applicationCache 进行需求实现。

参考链接：[HTML5 离线缓存-manifest 简介](https://yanhaijing.com/html/2014/12/28/html5-manifest/)

## 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。

## 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。
* 有效期（生命周期）：
  + 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)

* WebSocket、SharedWorker；
* 也可以调用 localstorge、cookies 等本地存储方式；

localstorge 另一个浏览上下文里被添加、修改或删除时，它都会触发一个事件，

我们通过监听事件，控制它的值来进行页面信息通信；

注意 quirks：Safari 在无痕模式下设置 localstorge 值时会抛出 QuotaExceededError 的异常；

## 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？

* 通过 visibilityState 的值检测页面当前是否可见，以及打开网页的时间等;
* 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？

* map+area 或者 svg
* border-radius
* 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:red"></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。

## title 与 h1 的区别、b 与 strong 的区别、i 与 em 的区别？

* title 属性没有明确意义只表示是个标题，H1 则表示层次明确的标题，对页面信息的抓取也有很大的影响；
* strong 是标明重点内容，有语气加强的含义，使用阅读设备阅读网络时：strong 会重读，而 b 是展示强调内容。
* i 内容展示为斜体，em 表示强调的文本；

Physical Style Elements -- 自然样式标签

b, i, u, s, pre

Semantic Style Elements -- 语义样式标签

strong, em, ins, del, code

应该准确使用语义样式标签, 但不能滥用, 如果不能确定时首选使用自然样式标签。

## img中 title 属性和 alt 属性的区别？

<img src="#" alt="alt信息" />

当图片不输出信息的时候，会显示 alt 信息 鼠标放上去没有信息，当图片正常读取，不会出现 alt 信息。

<img src="#" alt="alt信息" title="title信息" />

当图片不输出信息的时候，会显示 alt 信息 鼠标放上去会出现 title 信息； 当图片正常输出的时候，不会出现 alt 信息，鼠标放上去会出现 title 信息。

另外还有一些关于 title 属性的知识：

* title 属性可以用在除了 base，basefont，head，html，meta，param，script 和 title 之外的所有标签。
* title 属性的功能是提示。额外的说明信息和非本质的信息请使用 title 属性。title 属性值可以比 alt 属性值设置的更长。
* title 属性有一个很好的用途，即为链接添加描述性文字，特别是当连接本身并不是十分清楚的表达了链接的目的。

## 浏览器内核

**1，Trident**

Microsoft公司浏览器内核，IE6、IE7、IE8（Trident 4.0）、IE9（Trident 5.0）、IE10（Trident 6.0）及许多品牌浏览器的内核。其中部分浏览器的新版本是“双核”甚至是“多核”，其中一个内核是Trident，然后再增加一个其他内核。

**2，Gecko**

Firefox内核，Netscape6开始采用的内核，后来的Mozilla FireFox(火狐浏览器) ，Mozilla Firefox、Mozilla SeaMonkey、waterfox（Firefox的64位开源版）、Iceweasel、Epiphany（早期版本）、Flock（早期版本）、K-Meleon使用的内核。

**3，Presto**

Opera前内核，已废弃，Opera现已改用Google Chrome的Blink内核。

**4，Webkit**

Safari内核,Chrome内核原型,开源，它是苹果公司自己的内核，也是苹果的Safari浏览器使用的内核。 傲游浏览器3、Apple Safari、(Win/Mac/iPhone/iPad)、Symbian手机浏览器、Android 默认浏览器

**5，Blink**

Blink是一个由Google和Opera Software开发的浏览器排版引擎，Google计划将这个渲染引擎作为Chromium计划的一部分，这一渲染引擎是开源引擎WebKit中WebCore组件的一个分支，并且在Chrome（28及往后版本）、Opera（15及往后版本）。

**6，edge**

微软专门为新IE打造的引擎，速度快，目前已经基于此引擎开发了浏览器，目前IE11使用该内核，估计以后微软的新浏览器会继续采用该内核。

## 什么叫优雅降级和渐进增强？

**渐进增强 progressive enhancement：**  
针对低版本浏览器进行构建页面，保证最基本的功能，然后再针对高级浏览器进行效果、交互等改进和追加功能达到更好的用户体验。

**优雅降级 graceful degradation：**一开始就构建完整的功能，然后再针对低版本浏览器进行兼容。

**区别：**

a. 优雅降级是从复杂的现状开始，并试图减少用户体验的供给

b. 渐进增强则是从一个非常基础的，能够起作用的版本开始，并不断扩充，以适应未来环境的需要

c. 降级（功能衰减）意味着往回看；而渐进增强则意味着朝前看，同时保证其根基处于安全地带

# CSS

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

相同： 它们都能让元素不可见

区别：

* display:none;会让元素完全从渲染树中消失，渲染的时候不占据任何空间；visibility: hidden;不会让元素从渲染树消失，渲染师元素继续占据空间，只是内容不可见
* display: none;是非继承属性，子孙节点消失由于元素从渲染树消失造成，通过修改子孙节点属性无法显示；visibility:hidden;是继承属性，子孙节点消失由于继承了 hidden，通过设置 visibility: visible;可以让子孙节点显式
* 修改常规流中元素的 display 通常会造成文档重排。修改 visibility 属性只会造成本元素的重绘
* 读屏器不会读取 display: none;元素内容；会读取 visibility: hidden 元素内容

## css hack 原理及常用 hack

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

IE 条件注释：适用于[IE5, IE9]常见格式如下

<!--[if IE 6]>  
Special instructions for IE 6 here  
<![endif]-->

选择器 hack：不同浏览器对选择器的支持不一样

/\*\*\*\*\* Selector Hacks \*\*\*\*\*\*/  
​  
/\* 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+ \*/  
@media screen and (-webkit-min-device-pixel-ratio:0) {  
#diez { color: red }  
}  
​  
/\* iPhone / mobile webkit \*/  
@media screen and (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 \*/

## link 与 @import 的区别

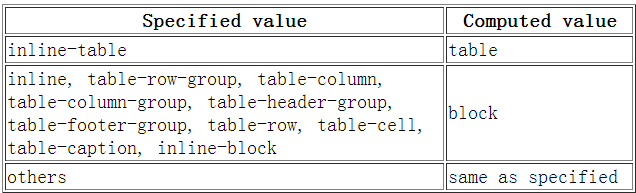
* link 是 HTML 方式， @import 是 CSS 方式
* link 最大限度支持并行下载，@import 过多嵌套导致串行下载，出现 FOUC
* link 可以通过 rel="alternate stylesheet" 指定候选样式
* 浏览器对 link 支持早于@import ，可以使用 @import 对老浏览器隐藏样式
* @import 必须在样式规则之前，可以在 css 文件中引用其他文件
* 总体来说：link 优于@import

## CSS 有哪些继承属性

* 关于文字排版的属性如：
  + font
  + word-break
  + letter-spacing
  + text-align
  + text-rendering
  + word-spacing
  + white-space
  + text-indent
  + text-transform
  + text-shadow
* line-height
* color
* visibility
* cursor

## display,float,position 的关系

* 如果 display 为 none，那么 position 和 float 都不起作用，这种情况下元素不产生框
* 否则，如果 position 值为 absolute 或者 fixed，框就是绝对定位的，float 的计算值为 none，display 根据下面的表格进行调整
* 否则，如果 float 不是 none，框是浮动的，display 根据下表进行调整
* 否则，如果元素是根元素，display 根据下表进行调整
* 其他情况下 display 的值为指定值 总结起来：绝对定位、浮动、根元素都需要调整display



## 外边距折叠(collapsing margins)

外边距重叠就是 margin-collapse

相邻的两个盒子（可能是兄弟关系也可能是祖先关系）的外边距可以结合成一个单独的外边距。 这种合并外边距的方式被称为折叠，结合而成的外边距称为折叠外边距

折叠结果遵循下列计算规则：

* 两个相邻的外边距都是正数时，折叠结果是它们两者之间较大的值
* 两个相邻的外边距都是负数时，折叠结果是两者绝对值的较大值
* 两个外边距一正一负时，折叠结果是两者的相加的和

## 介绍一下标准的 CSS 的盒子模型？低版本 IE 的盒子模型有什么不同的？

* 有两种， IE 盒子模型、W3C 盒子模型；
* 盒模型： 内容(content)、填充(padding)、边界(margin)、 边框(border)；
* 标准(W3C)盒模型：元素宽度 = width + padding + border + margin
* 怪异(IE)盒模型：元素宽度 = width + margin
* 区 别： IE 的 content 部分把 border 和 padding 计算了进去;
* 标准浏览器通过设置 css3 的 box-sizing: border-box 属性，触发“怪异模式”解析计算宽高

## CSS 选择符有哪些？

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

## CSS3 新增伪类有那些？

* 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。
* :after 在元素之前添加内容,也可以用来做清除浮动。
* :before 在元素之后添加内容
* :enabled 选择器匹配每个已启用的元素（大多用在表单元素上）。
* :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？

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

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

<body>  
   <div class="content">  
  aaaaaa aaaaaa a a a a a a a a  
   </div>  
</body>  
​  
<style>  
   body {  
       background: #DDD;  
       text-align: center; /\* 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>  
   <div class="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>  
   <div class="content">  
  aaaaaa aaaaaa a a a a a a a a  
   </div>  
</body>  
​  
<style>  
   body {  
       background: #DDD;  
       position: relative;  
  }  
   .content {  
       width: 800px;  
​  
       position: absolute;  
       left: 50%;  
       margin-left: -400px;  
​  
       background-color: purple;  
  }  
</style>

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

<body>  
   <div class="content">  
  aaaaaa aaaaaa a a a a a a a a  
   </div>  
</body>  
​  
<style>  
   body {  
       background: #DDD;  
       position: relative;  
  }  
   .content {  
       width: 800px;  
​  
       position: absolute;  
       margin: 0 auto;  
       left: 0;  
       right: 0;  
​  
       background-color: purple;  
  }  
</style>

## 如何竖直居中一个元素

* 绝对定位居中
* 如果居中的是行内元素，可以设置父级 height 与 line-height 相等
* 设置 margin/padding 居中
* 相对位置偏移居中
* flex 居中 设置 align-items:center 即可

## 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。
* table 此元素会作为块级表格来显示
* inherit 规定应该从父元素继承 display 属性的值

## position 有哪些值 relative 和 absolute 定位原点是？

* absolute 生成绝对定位的元素，相对于值不为 static 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？

* 新增选择器 p:nth-child(n){color: rgba(255, 0, 0, 0.75)}
* 弹性盒模型 display: flex;
* 多列布局 column-count: 5;
* 媒体查询 @media (max-width: 480px) {.box: {column-count: 1;}}
* 个性化字体 @font-face{font-family: BorderWeb; src:url(BORDERW0.eot);}
* 颜色透明度 color: rgba(255, 0, 0, 0.75);
* 圆角 border-radius: 5px;
* 渐变 background:linear-gradient(red, green, blue);
* 阴影 box-shadow:3px 3px 3px rgba(0, 64, 128, 0.3);
* 倒影 box-reflect: below 2px;
* 文字装饰 text-stroke-color: red;
* 文字溢出 text-overflow:ellipsis;
* 背景效果 background-size: 100px 100px;
* 边框效果 border-image:url(bt\_blue.png) 0 10;
* 平滑过渡 transition: all .3s ease-in .1s;
* 动画 @keyframes anim-1 {50% {border-radius: 50%;}} animation: anim-1 1s;
* 转换
  + 旋转 transform: rotate(20deg);
  + 倾斜 transform: skew(150deg, -10deg);
  + 位移 transform: translate(20px, 20px);
  + 缩放 transform: scale(.5);

## 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

## 经常遇到的浏览器的兼容性有哪些？原因，解决方法，常用 hack 的技巧 ？

* 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
* 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,event 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

## 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？

行框的排列会受到中间空白（回车\空格）等的影响，因为空格也属于字符,这些空白也会被应用样式，占据空间，所以会有间隔，把字符大小设为 0，就没有空格了

## display:inline-block 间隙问题怎么解决？(携程)

移除空格、使用 margin 负值、使用 font-size:0、letter-spacing、word-spacing

## display:inline-block 什么时候会显示间隙？

* 相邻的 inline-block 元素之间有换行或空格分隔的情况下会产生间距
* 非 inline-block 水平元素设置为 inline-block 也会有水平间距
* 可以借助 vertical-align:top; 消除垂直间隙
* 可以在父级加 font-size：0; 在子元素里设置需要的字体大小，消除垂直间隙
* 把 li 标签写到同一行可以消除垂直间隙，但代码可读性差

## css 定义的权重

网上有声称诸如id权重100，class权重10等计算方法，这是不正确的。 实际上应该如下：

1. 如果一个声明来自style属性而不是选择器，计作1或者a=1（在一个html文档中，元素“style”的值是样式表规则，这个规则中没有选择器，所以a=1, b=0, c=0, and d=0）
2. 选择器中id属性的个数,计作b
3. 选择器中其他属性以及伪类的个数，计作c
4. 选择器中元素及伪元素的个数，计作d

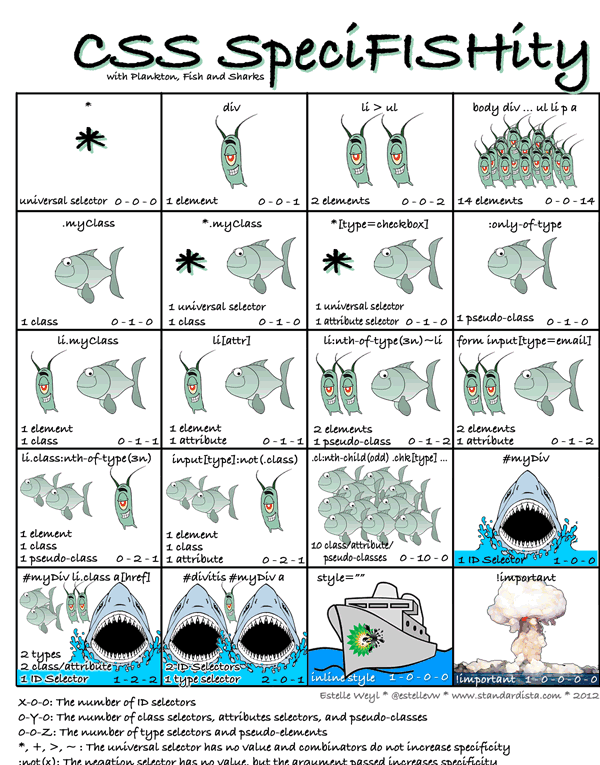
一些例子：

\* {}     /\* a=0 b=0 c=0 d=0 -> 优先级= 0,0,0,0 \*/  
li {}     /\* a=0 b=0 c=0 d=1 -> 优先级 = 0,0,0,1 \*/  
li:first-line {}     /\* a=0 b=0 c=0 d=2 -> 优先级 = 0,0,0,2 \*/  
ul li {}     /\* a=0 b=0 c=0 d=2 -> 优先级 = 0,0,0,2 \*/  
ul ol+li {}     /\* a=0 b=0 c=0 d=3 -> 优先级 = 0,0,0,3 \*/  
h1 + \*[rel=up]{}     /\* a=0 b=0 c=1 d=1 -> 优先级 = 0,0,1,1 \*/  
ul ol li.red {}     /\* a=0 b=0 c=1 d=3 -> 优先级 = 0,0,1,3 \*/  
li.red.level {}     /\* a=0 b=0 c=2 d=1 -> 优先级 = 0,0,2,1 \*/  
#x34y {}     /\* a=0 b=1 c=0 d=0 -> 优先级 = 0,1,0,0 \*/  
style=""     /\* a=1 b=0 c=0 d=0 -> 优先级 = 1,0,0,0 \*/  
​  
[备注]  
　　:first-line 伪元素  
　　[rel=up] 其他属性

优先级只基与选择器的形式，特殊的，一个“[id=p33]“形式的选择器是按照属性选择器来计算的（a=0, b=0, c=1, d=0），即使用定义中包含ID。

了解了这些 你应该不会再对”11个class与一个id”谁的优先级高“这类的问题有疑问了吧，因为a,b,c,d只是在各自位置数字的累加，而不会越级。

当然权重最高的是!important，会覆盖以上所有。行内样式也高不过它。

有一幅生动的图可以展示这个规则： 

## 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为准
* 优先级为: !important > id > class > tag important 比 内联优先级高

## 谈谈浮动和清除浮动

浮动的框可以向左或向右移动，直到他的外边缘碰到包含框或另一个浮动框的边框为止。由于浮动框不在文档的普通流中，所以文档的普通流的块框表现得就像浮动框不存在一样。浮动的块框会漂浮在文档普通流的块框上

解决方法

1. 父级 div 定义伪类：after 和 zoom (推荐使用，建议定义公共类，以减少 CSS 代码)

  .clearfloat:after{  
      display:block;  
      clear:both;  
      content:"";  
      visibility:hidden;  
      height:0}  
​  
  .clearfloat{zoom:1}

1. 在结尾处添加空 div 标签 clear:both

<div class="parent">  
   <div class="left">Left</div>  
   <div class="right">Right</div>  
   <div class="clearfloat"></div>  
</div>  
​  
<style>  
   .left {float:left}  
   .clearfloat{clear:both}  
</style>

1. 父级 div 定义 height
2. 父级 div 定义 overflow:auto
3. 父级 div 定义 overflow:hidden
4. 父级 div 也一起浮动
5. 父级 div 定义 display:table
6. 结尾处加 br 标签 clear:both

参考链接[几种常用的清除浮动方法](https://www.cnblogs.com/nxl0908/p/7245460.html)

## box-sizing 常用的属性有哪些？分别有什么作用？

* box-sizing: content-box; // 默认的标准(W3C)盒模型元素效果
* box-sizing: border-box; // 触发怪异(IE)盒模型元素的效果
* box-sizing: inherit; // 继承父元素 box-sizing 属性的值

## 请列举几种隐藏元素的方法

* visibility: hidden; 这个属性只是简单的隐藏某个元素，但是元素占用的空间任然存在
* opacity: 0; CSS3 属性，设置 0 可以使一个元素完全透明
* position: absolute; 设置一个很大的 left 负值定位，使元素定位在可见区域之外
* display: none; 元素会变得不可见，并且不会再占用文档的空间。
* transform: scale(0); 将一个元素设置为缩放无限小，元素将不可见，元素原来所在的位置将被保留
* <div hidden="hidden"> HTML5 属性,效果和 display:none;相同，但这个属性用于记录一个元素的状态
* height: 0; 将元素高度设为 0 ，并消除边框
* filter: blur(0); CSS3 属性，将一个元素的模糊度设置为 0，从而使这个

## rgba() 和 opacity 的透明效果有什么不同？

* opacity 作用于元素以及元素内的所有内容（包括文字）的透明度
* rgba() 只作用于元素自身的颜色或其背景色，子元素不会继承透明效果

## css 属性 content 有什么作用？

content 属性专门应用在 before/after 伪元素上，用于插入额外内容或样式

## 请解释一下 CSS3 的 Flexbox（弹性盒布局模型）以及适用场景？

Flexbox 用于不同尺寸屏幕中创建可自动扩展和收缩布局

## 请写出多种等高布局

* 在列的父元素上使用这个背景图进行 Y 轴的铺放，从而实现一种等高列的假像
* 模仿表格布局等高列效果：兼容性不好，在 ie6-7 无法正常运行
* css3 flexbox 布局： .container{display: flex; align-items: stretch;}

## 圣杯布局的实现原理？

要求：三列布局；中间主体内容前置，且宽度自适应；两边内容定宽

好处：重要的内容放在文档流前面可以优先渲染

原理：利用相对定位、浮动、负边距布局，而不添加额外标签

 .container {  
     padding-left: 150px;  
     padding-right: 190px;  
}  
 .main {  
     float: left;  
     width: 100%;  
}  
 .left {  
     float: left;  
     width: 190px;  
     margin-left: -100%;  
     position: relative;  
     left: -150px;  
}  
 .right {  
     float: left;  
     width: 190px;  
     margin-left: -190px;  
     position: relative;  
     right: -190px;  
}

## 什么是双飞翼布局？实现原理？

双飞翼布局：对圣杯布局（使用相对定位，对以后布局有局限性）的改进，消除相对定位布局

原理：主体元素上设置左右边距，预留两翼位置。左右两栏使用浮动和负边距归位，消除相对定位。

.container {  
   /\*padding-left:150px;\*/  
   /\*padding-right:190px;\*/  
}  
.main-wrap {  
   width: 100%;  
   float: left;  
}  
.main {  
   margin-left: 150px;  
   margin-right: 190px;  
}  
.left {  
   float: left;  
   width: 150px;  
   margin-left: -100%;  
   /\*position: relative;\*/  
   /\*left:-150px;\*/  
}  
.right {  
   float: left;  
   width: 190px;  
   margin-left: -190px;  
   /\*position:relative;\*/  
   /\*right:-190px;\*/  
}

## 在 CSS 样式中常使用 px、em 在表现上有什么区别？

* px 相对于显示器屏幕分辨率，无法用浏览器字体放大功能
* em 值并不是固定的，会继承父级的字体大小： em = 像素值 / 父级 font-size

## 为什么要初始化 CSS 样式？

* 不同浏览器对有些标签样式的默认值解析不同
* 不初始化 CSS 会造成各现浏览器之间的页面显示差异
* 可以使用 reset.css 或 Normalize.css 做 CSS 初始化

## reset.css 和 Normalize.css 理解

reset.css 意为重置默认样式。HTML 中绝大部分标签元素在网页显示中都有一个默认属性值，通常为了避免重复定义元素样式，需要进行重置默认样式

Eric Meyer（CSS Reset）推荐

html, body, div, span, applet, object, iframe,  
h1, h2, h3, h4, h5, h6, p, blockquote, pre,  
a, abbr, acronym, address, big, cite, code,  
del, dfn, em, font, img, ins, kbd, q, s, samp,  
small, strike, strong, sub, sup, tt, var,  
b, u, i, center,  
dl, dt, dd, ol, ul, li,  
fieldset, form, label, legend,  
table, caption, tbody, tfoot, thead, tr, th, td {  
   margin: 0;  
   padding: 0;  
   border: 0;  
   outline: 0;  
   font-size: 100%;  
   vertical-align: baseline;  
   background: transparent;  
}  
body {  
   line-height: 1;  
}  
ol, ul {  
   list-style: none;  
}  
blockquote, q {  
   quotes: none;  
}  
blockquote:before, blockquote:after,  
q:before, q:after {  
   content: '';  
   content: none;  
}  
/\* remember to define focus styles! \*/  
:focus {  
   outline: 0;  
}  
/\* remember to highlight inserts somehow! \*/  
ins {  
   text-decoration: none;  
}  
del {  
   text-decoration: line-through;  
}  
/\* tables still need ‘cellspacing=”0″‘ in the markup \*/  
table {  
   border-collapse: collapse;  
   border-spacing: 0;  
}

Normalize.css 只是一个很小的 css 文件,但它在默认的 HTML 元素样式上提供了跨浏览器的高度一致性。相比于传统的 css reset，Normalize.css 是一种现代的，为 HTML5 准备的优质替代方案。

Normalize.css 是一种 CSS reset 的替代方案。经过@necolas 和@jon neal 花了几百个小时来努力研究不同浏览器的默认样式的差异，这个项目终于变成了现在这样。

我们创造 normalize.css 有下几个目的：

* 保护有用的浏览器默认样式而不是完全去掉它们
* 一般化的样式：为大部分 HTML 元素提供
* 修复浏览器自身的 bug 并保证各浏览器的一致性
* 优化 CSS 可用性：用一些小技巧
* 解释代码：用注释和详细的文档来

## 什么是 FOUC(Flash of Unstyled Content)？ 如何来避免 FOUC？

* 当使用 @import 导入 CSS 时，会导致某些页面在 IE 出现奇怪的现象： 没有样式的页面内容显示瞬间闪烁，这种现象称为“文档样式短暂失效”，简称为 FOUC
* 产生原因：当样式表晚于结构性 html 加载时，加载到此样式表时，页面将停止之前的渲染。
* 等待此样式表被下载和解析后，再重新渲染页面，期间导致短暂的花屏现象。
* 解决方法：使用 link 标签将样式表放在文档 head

## 介绍使用过的 CSS 预处理器？

* CSS 预处理器基本思想：为 CSS 增加了一些编程的特性（变量、逻辑判断、函数等）
* 开发者使用这种语言进行进行 Web 页面样式设计，再编译成正常的 CSS 文件使用
* 使用 CSS 预处理器，可以使 CSS 更加简洁、适应性更强、可读性更佳，无需考虑兼容性
* 最常用的 CSS 预处理器语言包括：Sass（SCSS）和 LESS

## CSS 优化、提高性能的方法有哪些？

* 多个 css 合并，尽量减少 HTTP 请求
* 将 css 文件放在页面最上面
* 移除空的 css 规则
* 避免使用 CSS 表达式
* 选择器优化嵌套，尽量避免层级过深
* 充分利用 css 继承属性，减少代码量
* 抽象提取公共样式，减少代码量
* 属性值为 0 时，不加单位
* 属性值为小于 1 的小数时，省略小数点前面的 0
* css 雪碧图

## 浏览器是怎样解析 CSS 选择器的？

浏览器解析 CSS 选择器的方式是从右到左

## 在网页中的应该使用奇数还是偶数的字体？

在网页中的应该使用“偶数”字体：

* 偶数字号相对更容易和 web 设计的其他部分构成比例关系
* 使用奇数号字体时文本段落无法对齐
* 宋体的中文网页排布中使用最多的就是 12 和 14

## margin 和 padding 分别适合什么场景使用？

* 需要在 border 外侧添加空白，且空白处不需要背景（色）时，使用 margin
* 需要在 border 内测添加空白，且空白处需要背景（色）时，使用 padding

## 抽离样式模块怎么写，说出思路？

CSS 可以拆分成 2 部分：公共 CSS 和 业务 CSS：

* 网站的配色，字体，交互提取出为公共 CSS。这部分 CSS 命名不应涉及具体的业务
* 对于业务 CSS，需要有统一的命名，使用公用的前缀。可以参考面向对象的 CSS

## 元素竖向的百分比设定是相对于容器的高度吗？

元素竖向的百分比设定是相对于容器的宽度，而不是高度

## 全屏滚动的原理是什么？ 用到了 CSS 的那些属性？

* 原理类似图片轮播原理，超出隐藏部分，滚动时显示
* 可能用到的 CSS 属性：overflow:hidden; transform:translate(100%, 100%); display:none;

## 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？

* 响应式设计就是网站能够兼容多个终端，而不是为每个终端做一个特定的版本
* 基本原理是利用 CSS3 媒体查询，为不同尺寸的设备适配不同样式
* 对于低版本的 IE，可采用 JS 获取屏幕宽度，然后通过 resize 方法来实现兼容：

$(window).resize(function () {  
   screenRespond();  
});  
​  
screenRespond();  
​  
function screenRespond(){  
​  
   var screenWidth = $(window).width();  
​  
   if(screenWidth <= 1800){  
       $("body").attr("class", "w1800");  
  }  
​  
   if(screenWidth <= 1400){  
       $("body").attr("class", "w1400");  
  }  
​  
   if(screenWidth > 1800){  
       $("body").attr("class", "");  
  }  
}

## 什么是视差滚动效果，如何给每页做不同的动画？

* 视差滚动是指多层背景以不同的速度移动，形成立体的运动效果，具有非常出色的视觉体验
* 一般把网页解剖为：背景层、内容层和悬浮层。当滚动鼠标滚轮时，各图层以不同速度移动，形成视差的

实现原理

* 以 “页面滚动条” 作为 “视差动画进度条”
* 以 “滚轮刻度” 当作 “动画帧度” 去播放动画的
* 监听 mousewheel 事件，事件被触发即播放动画，实现“翻页”效果

## a 标签上四个伪类的执行顺序是怎么样的？

link > visited > hover > active

## 伪元素和伪类的区别和作用？

伪元素:在内容元素的前后插入额外的元素或样式，但是这些元素实际上并不在文档中生成。它们只在外部显示可见，但不会在文档的源代码中找到它们，因此，称为“伪”元素。例如：

p::before {content:"第一章：";}  
p::after {content:"Hot!";}  
p::first-line {background:red;}  
p::first-letter {font-size:30px;}

伪类: 将特殊的效果添加到特定选择器上。它是已有元素上添加类别的，不会产生新的元素。例如：

a:hover {color: #FF00FF}  
p:first-child {color: red}

## ::before 和 :after 中双冒号和单冒号有什么区别？

* 在 CSS 中伪类一直用 : 表示，如 :hover, :active 等
* 伪元素在 CSS1 中已存在，当时语法是用 : 表示，如 :before 和 :after
* 后来在 CSS3 中修订，伪元素用 :: 表示，如 ::before 和 ::after，以此区分伪元素和伪类
* 由于低版本 IE 对双冒号不兼容，开发者为了兼容性各浏览器，继续使使用 :after 这种老语法表示伪元素
* 综上所述：::before 是 CSS3 中写伪元素的新语法； :after 是 CSS1 中存在的、兼容 IE 的老语法

## 如何修改 Chrome 记住密码后自动填充表单的黄色背景？

* 产生原因：由于 Chrome 默认会给自动填充的 input 表单加上 input:-webkit-autofill 私有属性造成的
* 解决方案 1：在 form 标签上直接关闭了表单的自动填充：autocomplete="off"
* 解决方案 2：input:-webkit-autofill { background-color: transparent; }

input [type=search] 搜索框右侧小图标如何美化？

input[type="search"]::-webkit-search-cancel-button{  
 -webkit-appearance: none;  
 height: 15px;  
 width: 15px;  
 border-radius: 8px;  
 background:url("images/searchicon.png") no-repeat 0 0;  
 background-size: 15px 15px;  
}

## 网站图片文件，如何点击下载？而非点击预览？

<a href="logo.jpg" download>下载</a> <a href="logo.jpg" download="网站LOGO" >下载</a>

## iOS safari 如何阻止“橡皮筋效果”？

 $(document).ready(function(){  
     var stopScrolling = function(event) {  
         event.preventDefault();  
    }  
     document.addEventListener('touchstart', stopScrolling, false);  
     document.addEventListener('touchmove', stopScrolling, false);  
});

## 你对 line-height 是如何理解的？

* line-height 指一行字的高度，包含了字间距，实际上是下一行基线到上一行基线距离
* 如果一个标签没有定义 height 属性，那么其最终表现的高度是由 line-height 决定的
* 一个容器没有设置高度，那么撑开容器高度的是 line-height 而不是容器内的文字内容
* 把 line-height 值设置为 height 一样大小的值可以实现单行文字的垂直居中
* line-height 和 height 都能撑开一个高度，height 会触发 haslayout，而 line-height 不会

## line-height 三种赋值方式有何区别？（带单位、纯数字、百分比）

* 带单位：px 是固定值，而 em 会参考父元素 font-size 值计算自身的行高
* 纯数字：会把比例传递给后代。例如，父级行高为 1.5，子元素字体为 18px，则子元素行高为 1.5 \* 18 = 27px
* 百分比：将计算后的值传递给后代

## 设置元素浮动后，该元素的 display 值会如何变化？

设置元素浮动后，该元素的 display 值自动变成 block

## 怎么让 Chrome 支持小于 12px 的文字？

 .shrink{  
   -webkit-transform:scale(0.8);  
   -o-transform:scale(1);  
   display:inline-block;  
}

## 让页面里的字体变清晰，变细用 CSS 怎么做？（IOS 手机浏览器字体齿轮设置）

 -webkit-font-smoothing: antialiased;

font-style 属性 oblique 是什么意思？

font-style: oblique; 使没有 italic 属性的文字实现倾斜

## 如果需要手动写动画，你认为最小时间间隔是多久？

16.7ms 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: 1s / 60 \* 1000 ＝ 16.7ms

## overflow: scroll 时不能平滑滚动的问题怎么处理？

监听滚轮事件，然后滚动到一定距离时用 jquery 的 animate 实现平滑效果。

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

* 方案 1： .sub { height: calc(100%-100px); }
* 方案 2： .container { position:relative; } .sub { position: absolute; top: 100px; bottom: 0; }
* 方案 3： .container { display:flex; flex-direction:column; } .sub { flex:1; }

## CSS 中类 class 和 id 的区别

对于 CSS 而言，id 和 class 都是选择器，唯一不同的地方在于权重不同。如果只说 CSS，上面那一句话就讲完了。拓展出来，对于 html 而言，id 和 class 都是 dom 元素的属性值。不同的地方在于 id 属性的值是唯一的，而 class 属性值可以重复。id 还一个老特性是锚点功能，当浏览器地址栏有一个#xxx，页面会自动滚动到 id=xxx 的元素上面。

更直接的：id 给 js 用，class 给 css 用（趋势）

## 如何优化网页的打印样式

<link rel="stylesheet" type="text/css" media="screen" href="xxx.css" />

其中 media 指定的属性就是设备，显示器上就是 screen，打印机则是 print，电视是 tv，投影仪是 projection。

<link rel="stylesheet" type="text/css" media="print" href="yyy.css" />

但打印样式表也应有些注意事项：

* 打印样式表中最好不要用背景图片，因为打印机不能打印 CSS 中的背景。如要显示图片，请使用 html 插入到页面中。
* 最好不要使用像素作为单位，因为打印样式表要打印出来的会是实物，所以建议使用 pt 和 cm。
* 隐藏掉不必要的内容。（@print div{display:none;}）
* 打印样式表中最好少用浮动属性，因为它们会消失。

## 请问为何要使用 transform 而非 absolute positioning，或反之的理由？为什么？

* 使用 transform 或 position 实现动画效果时是有很大差别。
* 使用 transform 时，可以让 GPU 参与运算，动画的 FPS 更高。
* 使用 position 时，最小的动画变化的单位是 1px，而使用 transform 参与时，可以做到更小（动画效果更加平滑）
* 功能都一样。但是 translate 不会引起浏览器的重绘和重排，这就相当 nice 了。

反之

* tranform 改变 fixed 子元素的定位对象
* transform 改变元素层叠顺序 [transform 的副作用](http://imweb.io/topic/5a23e1f1a192c3b460fce26e)

## 请解释 CSS sprites，以及你要如何在页面或网站中实现它

* CSS Sprites 其实就是把网页中一些背景图片整合到一张图片文件中，再利用 CSS 的“background-image”，“background- repeat”，“background-position”的组合进行背景定位，background-position 可以用数字能精确的定位出背景图片的位置。
* CSS Sprites 为一些大型的网站节约了带宽，让提高了用户的加载速度和用户体验，不需要加载更多的图片。

## 你熟悉 SVG 样式的书写吗？

| **SVG** | **等效的 CSS** |
| --- | --- |
| fill | background-color 或 color |
| fill-opacity | background-color 或 color 设置 rgba/hsla |
| opacity | opacity |
| stroke | border-color |
| stroke-width | border-thickness |
| stroke-opacity | border-color 设置 rgba |
| rx, ry | border-radius |

下面的属性在 SVG 和 CSS 中是一样的，只是在 SVG 的 transformations 和 dimensions 中稍有区别：

* font-family, font-size, font-style, font-variant 和 font-weight
* width 和 height
* scale, rotate, skew

参考链接： [基本的 SVG 样式属性](http://justcode.ikeepstudying.com/2016/08/%E5%9F%BA%E6%9C%AC%E7%9A%84svg%E6%A0%B7%E5%BC%8F%E5%B1%9E%E6%80%A7/)

## 如果设计中使用了非标准的字体，你该如何去实现？

* 用图片代替
* web fonts 在线字库
* @font-face

参考链接：[如果设计中使用了非标准的字体，你该如何去实现？](https://blog.csdn.net/xujie_0311/article/details/42368371)

# JavaScript基础

## JavaScript 的组成

JavaScript 由以下三部分组成：

* ECMAScript（核心）：JavaScript 语言基础
* DOM（文档对象模型）：规定了访问 HTML 和 XML 的接口
* BOM（浏览器对象模型）：提供了浏览器窗口之间进行交互的对象和方法

## JS 的基本数据类型和引用数据类型

* 基本数据类型：undefined、null、boolean、number、string、symbol
* 引用数据类型：object、array、function

## 检测浏览器版本版本有哪些方式？

* 根据 navigator.userAgent // UA.toLowerCase().indexOf('chrome')
* 根据 window 对象的成员 // 'ActiveXObject' in window

## 介绍 JS 有哪些内置对象？

* 数据封装类对象：Object、Array、Boolean、Number、String
* 其他对象：Function、Arguments、Math、Date、RegExp、Error
* ES6 新增对象：Symbol、Map、Set、Promises、Proxy、Reflect

## 说几条写 JavaScript 的基本规范？

* 代码缩进，建议使用“四个空格”缩进
* 代码段使用花括号{}包裹
* 语句结束使用分号;
* 变量和函数在使用前进行声明
* 以大写字母开头命名构造函数，全大写命名常量
* 规范定义 JSON 对象，补全双引号
* 用{}和[]声明对象和数组

## 如何编写高性能的 JavaScript？

* 遵循严格模式："use strict";
* 将 js 脚本放在页面底部，加快渲染页面
* 将 js 脚本将脚本成组打包，减少请求
* 使用非阻塞方式下载 js 脚本
* 尽量使用局部变量来保存全局变量
* 尽量减少使用闭包
* 使用 window 对象属性方法时，省略 window
* 尽量减少对象成员嵌套
* 缓存 DOM 节点的访问
* 通过避免使用 eval() 和 Function() 构造器
* 给 setTimeout() 和 setInterval() 传递函数而不是字符串作为参数
* 尽量使用直接量创建对象和数组
* 最小化重绘(repaint)和回流(reflow)

## 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

## 描述浏览器的渲染过程，DOM 树和渲染树的区别？

浏览器的渲染过程：

* 解析 HTML 构建 DOM(DOM 树)，并行请求 css/image/js
* CSS 文件下载完成，开始构建 CSSOM(CSS 树)
* CSSOM 构建结束后，和 DOM 一起生成 Render Tree(渲染树)
* 布局(Layout)：计算出每个节点在屏幕中的位置
* 显示(Painting)：通过显卡把页面画到屏幕上

DOM 树 和 渲染树 的区别：

* DOM 树与 HTML 标签一一对应，包括 head 和隐藏元素
* 渲染树不包括 head 和隐藏元素，大段文本的每一个行都是独立节点，每一个节点都有对应的 css 属性

## 重绘和回流（重排）的区别和关系？

* 重绘：当渲染树中的元素外观（如：颜色）发生改变，不影响布局时，产生重绘
* 回流：当渲染树中的元素的布局（如：尺寸、位置、隐藏/状态状态）发生改变时，产生重绘回流
* 注意：JS 获取 Layout 属性值（如：offsetLeft、scrollTop、getComputedStyle 等）也会引起回流。因为浏览器需要通过回流计算最新值
* 回流必将引起重绘，而重绘不一定会引起回流

## 如何最小化重绘(repaint)和回流(reflow)？

* 需要要对元素进行复杂的操作时，可以先隐藏(display:"none")，操作完成后再显示
* 需要创建多个 DOM 节点时，使用 DocumentFragment 创建完后一次性的加入 document
* 缓存 Layout 属性值，如：var left = elem.offsetLeft; 这样，多次使用 left 只产生一次回流
* 尽量避免用 table 布局（table 元素一旦触发回流就会导致 table 里所有的其它元素回流）
* 避免使用 css 表达式(expression)，因为每次调用都会重新计算值（包括加载页面）
* 尽量使用 css 属性简写，如：用 border 代替 border-width, border-style, border-color 批量修改元素样式：elem.className 和 elem.style.cssText 代替 elem.style.xxx

## script 的位置是否会影响首屏显示时间？

* 在解析 HTML 生成 DOM 过程中，js 文件的下载是并行的，不需要 DOM 处理到 script 节点。因此，script 的位置不影响首屏显示的开始时间。
* 浏览器解析 HTML 是自上而下的线性过程，script 作为 HTML 的一部分同样遵循这个原则
* 因此，script 会延迟 DomContentLoad，只显示其上部分首屏内容，从而影响首屏显示的完成时间

## 解释 JavaScript 中的作用域与变量声明提升？

JavaScript 作用域：

* 在 Java、C 等语言中，作用域为 for 语句、if 语句或{}内的一块区域，称为作用域；
* 而在 JavaScript 中，作用域为 function(){}内的区域，称为函数作用域。

JavaScript 变量声明提升：

* 在 JavaScript 中，函数声明与变量声明经常被 JavaScript 引擎隐式地提升到当前作用域的顶部。
* 声明语句中的赋值部分并不会被提升，只有名称被提升
* 函数声明的优先级高于变量，如果变量名跟函数名相同且未赋值，则函数声明会覆盖变量声明
* 如果函数有多个同名参数，那么最后一个参数（即使没有定义）会覆盖前面的同名参数

## 介绍 JavaScript 的原型，原型链？有什么特点？

原型：

* JavaScript 的所有对象中都包含了一个 [proto] 内部属性，这个属性所对应的就是该对象的原型
* JavaScript 的函数对象，除了原型 [proto] 之外，还预置了 prototype 属性
* 当函数对象作为构造函数创建实例时，该 prototype 属性值将被作为实例对象的原型 [proto]。

原型链：

* 当一个对象调用的属性/方法自身不存在时，就会去自己 [proto] 关联的前辈 prototype 对象上去找
* 如果没找到，就会去该 prototype 原型 [proto] 关联的前辈 prototype 去找。依次类推，直到找到属性/方法或 undefined 为止。从而形成了所谓的“原型链”

原型特点：

* JavaScript 对象是通过引用来传递的，当修改原型时，与之相关的对象也会继承这一改变

## JavaScript 有几种类型的值？你能画一下他们的内存图吗

* 原始数据类型（Undefined，Null，Boolean，Number、String）-- 栈
* 引用数据类型（对象、数组和函数）-- 堆
* 两种类型的区别是：存储位置不同：
* 原始数据类型是直接存储在栈(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；
* 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 如何实现一个类，怎么实例化这个类？

1. 构造函数法（this + prototype） -- 用 new 关键字 生成实例对象
   * 缺点：用到了 this 和 prototype，编写复杂，可读性差

 function Mobile(name, price){  
    this.name = name;  
    this.price = price;  
  }  
  Mobile.prototype.sell = function(){  
     alert(this.name + "，售价 $" + this.price);  
  }  
  var iPhone7 = new Mobile("iPhone7", 1000);  
  iPhone7.sell();

1. Object.create 法 -- 用 Object.create() 生成实例对象
   * 缺点：不能实现私有属性和私有方法，实例对象之间也不能共享数据

var Person = {  
    firstname: "Mark",  
    lastname: "Yun",  
    age: 25,  
    introduce: function(){  
        alert('I am ' + Person.firstname + ' ' + Person.lastname);  
    }  
};  
​  
var person = Object.create(Person);  
person.introduce();  
​  
// Object.create 要求 IE9+，低版本浏览器可以自行部署：  
if (!Object.create) {  
　   Object.create = function (o) {  
　　　 function F() {}  
　　　 F.prototype = o;  
　　　 return new F();  
　　};  
　}

1. 极简主义法（消除 this 和 prototype） -- 调用 createNew() 得到实例对象
   * 优点：容易理解，结构清晰优雅，符合传统的"面向对象编程"的构造

var Cat = {  
  age: 3, // 共享数据 -- 定义在类对象内，createNew() 外  
  createNew: function () {  
    var cat = {};  
    // var cat = Animal.createNew(); // 继承 Animal 类  
    cat.name = "小咪";  
    var sound = "喵喵喵"; // 私有属性--定义在 createNew() 内，输出对象外  
    cat.makeSound = function () {  
      alert(sound);  // 暴露私有属性  
    };  
    cat.changeAge = function(num){  
      Cat.age = num; // 修改共享数据  
    };  
    return cat; // 输出对象  
  }  
};  
​  
var cat = Cat.createNew();  
cat.makeSound();

1. ES6 语法糖 class -- 用 new 关键字 生成实例对象

    class Point {  
      constructor(x, y) {  
        this.x = x;  
        this.y = y;  
      }  
      toString() {  
        return '(' + this.x + ', ' + this.y + ')';  
      }  
    }  
​  
 var point = new Point(2, 3);

## Javascript 如何实现继承？

1. 构造函数绑定：使用 call 或 apply 方法，将父对象的构造函数绑定在子对象上

function Cat(name,color){  
　Animal.apply(this, arguments);  
　this.name = name;  
　this.color = color;  
}

1. 实例继承：将子对象的 prototype 指向父对象的一个实例

Cat.prototype = new Animal();  
Cat.prototype.constructor = Cat;

1. 拷贝继承：如果把父对象的所有属性和方法，拷贝进子对象

function extend(Child, Parent) {  
　　　var p = Parent.prototype;  
　　　var c = Child.prototype;  
　　　for (var i in p) {  
　　　   c[i] = p[i];  
　　　}  
　　　c.uber = p;  
}

1. 原型继承：将子对象的 prototype 指向父对象的 prototype

function extend(Child, Parent) {  
   var F = function(){};  
  　F.prototype = Parent.prototype;  
  　Child.prototype = new F();  
  　Child.prototype.constructor = Child;  
  　Child.uber = Parent.prototype;  
}

1. ES6 语法糖 extends：class ColorPoint extends Point {}

class ColorPoint extends Point {  
   constructor(x, y, color) {  
       super(x, y); // 调用父类的constructor(x, y)  
       this.color = color;  
  }  
   toString() {  
       return this.color + ' ' + super.toString(); // 调用父类的toString()  
  }  
}

## js 继承方式及其优缺点

原型链继承的缺点

* 一是字面量重写原型会中断关系，使用引用类型的原型，并且子类型还无法给超类型传递参数。

借用构造函数（类式继承）

* 借用构造函数虽然解决了刚才两种问题，但没有原型，则复用无从谈起。所以我们需要原型链+借用构造函数的模式，这种模式称为组合继承

组合式继承

* 组合式继承是比较常用的一种继承方法，其背后的思路是使用原型链实现对原型属性和方法的继承，而通过借用构造函数来实现对实例属性的继承。这样，既通过在原型上定义方法实现了函数复用，又保证每个实例都有它自己的属性。

## 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"};

1. 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();

1. 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

1. 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();

1. 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();

1. 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();

## eval 是做什么的？

eval 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（先解析成 js 语句，再执行）
* 由 JSON 字符串转换为 JSON 对象的时候可以用 eval('('+ str +')');

## 事件的三个阶段

捕获、目标、冒泡

## 介绍事件“捕获”和“冒泡”执行顺序和事件的执行次数？

按照 W3C 标准的事件：首是进入捕获阶段，直到达到目标元素，再进入冒泡阶段

事件执行次数（DOM2-addEventListener）：元素上绑定事件的个数

* 注意 1：前提是事件被确实触发
* 注意 2：事件绑定几次就算几个事件，即使类型和功能完全一样也不会“覆盖”

事件执行顺序：判断的关键是否目标元素

* 非目标元素：根据 W3C 的标准执行：捕获->目标元素->冒泡（不依据事件绑定顺序）
* 目标元素：依据事件绑定顺序：先绑定的事件先执行（不依据捕获冒泡标准）
* 最终顺序：父元素捕获->目标元素事件 1->目标元素事件 2->子元素捕获->子元素冒泡->父元素冒泡
* 注意：子元素事件执行前提 事件确实“落”到子元素布局区域上，而不是简单的具有嵌套关系

## 在一个 DOM 上同时绑定两个点击事件：一个用捕获，一个用冒泡。事件会执行几次，先执行冒泡还是捕获？

* 该 DOM 上的事件如果被触发，会执行两次（执行次数等于绑定次数）
* 如果该 DOM 是目标元素，则按事件绑定顺序执行，不区分冒泡/捕获
* 如果该 DOM 是处于事件流中的非目标元素，则先执行捕获，后执行冒泡

## 事件的代理/委托

事件委托是指将事件绑定目标元素的到父元素上，利用冒泡机制触发该事件

优点：

* 可以减少事件注册，节省大量内存占用
* 可以将事件应用于动态添加的子元素上

缺点： 使用不当会造成事件在不应该触发时触发

示例：

ulEl.addEventListener('click', function(e){  
   var target = event.target || event.srcElement;  
   if(!!target && target.nodeName.toUpperCase() === "LI"){  
       console.log(target.innerHTML);  
  }  
}, false);

## IE 与火狐的事件机制有什么区别？ 如何阻止冒泡？

IE 只事件冒泡，不支持事件捕获；火狐同时支持件冒泡和事件捕获。

阻止冒泡：

* 取消默认操作: w3c 的方法是 e.preventDefault()，IE 则是使用 e.returnValue = false;
* return false javascript 的 return false 只会阻止默认行为，而是用 jQuery 的话则既阻止默认行为又防止对象冒泡。
* 阻止冒泡 w3c 的方法是 e.stopPropagation()，IE 则是使用 e.cancelBubble = true

[js] view plaincopy  
function stopHandler(event)​  
   window.event?window.event.cancelBubble=true:event.stopPropagation();​  
}

## IE 的事件处理和 W3C 的事件处理有哪些区别？(必考)

绑定事件

* W3C: targetEl.addEventListener('click', handler, false);
* IE: targetEl.attachEvent('onclick', handler);

删除事件

* W3C: targetEl.removeEventListener('click', handler, false);
* IE: targetEl.detachEvent(event, handler);

事件对象

* W3C: var e = arguments.callee.caller.arguments[0]
* IE: window.event

事件目标

* W3C: e.target
* IE: window.event.srcElement

阻止事件默认行为

* W3C: e.preventDefault()
* IE: window.event.returnValue = false'

阻止事件传播

* W3C: e.stopPropagation()
* IE: window.event.cancelBubble = true

## W3C 事件的 target 与 currentTarget 的区别？

* target 只会出现在事件流的目标阶段
* currentTarget 可能出现在事件流的任何阶段
* 当事件流处在目标阶段时，二者的指向相同
* 当事件流处于捕获或冒泡阶段时：currentTarget 指向当前事件活动的对象(一般为父级)

## 什么是函数节流？介绍一下应用场景和原理？

* 函数节流(throttle)是指阻止一个函数在很短时间间隔内连续调用。 只有当上一次函数执行后达到规定的时间间隔，才能进行下一次调用。 但要保证一个累计最小调用间隔（否则拖拽类的节流都将无连续效果）
* 函数节流用于 onresize, onscroll 等短时间内会多次触发的事件
* 函数节流的原理：使用定时器做时间节流。 当触发一个事件时，先用 setTimout 让这个事件延迟一小段时间再执行。 如果在这个时间间隔内又触发了事件，就 clearTimeout 原来的定时器， 再 setTimeout 一个新的定时器重复以上流程。

函数节流简单实现：

function throttle(method, context) {  
    clearTimeout(methor.tId);  
    method.tId = setTimeout(function(){  
        method.call(context);  
    }， 100); // 两次调用至少间隔 100ms  
}  
// 调用  
window.onresize = function(){  
   throttle(myFunc, window);  
}

## 区分什么是“客户区坐标”、“页面坐标”、“屏幕坐标”？

* 客户区坐标：鼠标指针在可视区中的水平坐标(clientX)和垂直坐标(clientY)
* 页面坐标：鼠标指针在页面布局中的水平坐标(pageX)和垂直坐标(pageY)
* 屏幕坐标：设备物理屏幕的水平坐标(screenX)和垂直坐标(screenY)

## 如何获得一个 DOM 元素的绝对位置？

* elem.offsetLeft：返回元素相对于其定位父级左侧的距离
* elem.offsetTop：返回元素相对于其定位父级顶部的距离
* elem.getBoundingClientRect()：返回一个 DOMRect 对象，包含一组描述边框的只读属性，单位像素

## 分析 ['1', '2', '3'].map(parseInt) 答案是多少？（常考）

答案:[1, NaN, NaN]

parseInt(string, radix) 第 2 个参数 radix 表示进制。省略 radix 或 radix = 0，则数字将以十进制解析

map 每次为 parseInt 传 3 个参数(elem, index, array)，其中 index 为数组索引

因此，map 遍历 ["1", "2", "3"]，相应 parseInt 接收参数如下

parseInt('1', 0);  // 1  
parseInt('2', 1);  // NaN  
parseInt('3', 2);  // NaN

所以，parseInt 参数 radix 不合法，导致返回值为 NaN

## new 操作符具体干了什么？

* 创建实例对象，this 变量引用该对象，同时还继承了构造函数的原型
* 属性和方法被加入到 this 引用的对象中
* 新创建的对象由 this 所引用，并且最后隐式的返回 this

## 用原生 JavaScript 的实现过什么功能吗？

封装选择器、调用第三方 API、设置和获取样式(自由回答)

解释一下这段代码的意思吗？

[].forEach.call($$("\*"), function(el){  
     el.style.outline = "1px solid #" + (~~(Math.random()\*(1<<24))).toString(16);  
})

解释：获取页面所有的元素，遍历这些元素，为它们添加 1 像素随机颜色的轮廓(outline)

* ​函数被许多现代浏览器命令行支持，等价于 document.querySelectorAll(sel)
* [].forEach.call(NodeLists) // 使用 call 函数将数组遍历函数 forEach 应到节点元素列表
* el.style.outline = "1px solid #333" // 样式 outline 位于盒模型之外，不影响元素布局位置
* (1<<24) // parseInt("ffffff", 16) == 16777215 == 2^24 - 1 // 1<<24 == 2^24 == 16777216
* Math.random()\*(1<<24) // 表示一个位于 0 到 16777216 之间的随机浮点数
* ~~Math.random()\*(1<<24) // ~~ 作用相当于 parseInt 取整
* (~~(Math.random()\*(1<<24))).toString(16) // 转换为一个十六进制-

## 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

闭包的特性：

* 函数内再嵌套函数
* 内部函数可以引用外层的参数和变量
* 参数和变量不会被垃圾回收机制回收

## 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

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');  
  }

## 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

## defer 和 async

defer 并行加载 js 文件，会按照页面上 script 标签的顺序执行 async 并行加载 js 文件，下载完成立即执行，不会按照页面上 script 标签的顺序执行

## 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,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

## 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

## 说说你对闭包的理解

使用闭包主要是为了设计私有的方法和变量。闭包的优点是可以避免全局变量的污染，缺点是闭包会常驻内存，会增大内存使用量，使用不当很容易造成内存泄露。在 js 中，函数即闭包，只有函数才会产生作用域的概念

闭包有三个特性：

* 函数嵌套函数
* 函数内部可以引用外部的参数和变量
* 参数和变量不会被垃圾回收机制回收

## 请解释一下 JavaScript 的同源策略

* 概念:同源策略是客户端脚本（尤其是 Javascript）的重要的安全度量标准。它最早出自 Netscape Navigator2.0，其目的是防止某个文档或脚本从多个不同源装载。这里的同源策略指的是：协议，域名，端口相同，同源策略是一种安全协议
* 指一段脚本只能读取来自同一来源的窗口和文档的属性

## 为什么要有同源限制？

我们举例说明：比如一个黑客程序，他利用 Iframe 把真正的银行登录页面嵌到他的页面上，当你使用真实的用户名，密码登录时，他的页面就可以通过 Javascript 读取到你的表单中 input 中的内容，这样用户名，密码就轻松到手了。]

缺点: 现在网站的 JS 都会进行压缩，一些文件用了严格模式，而另一些没有。这时这些本来是严格模式的文件，被 merge 后，这个串就到了文件的中间，不仅没有指示严格模式，反而在压缩后浪费了字节

实现一个函数 clone，可以对 JavaScript 中的 5 种主要的数据类型（包括 Number、String、Object、Array、Boolean）进行值复制（常考）

function deepClone(obj) {  
   if (!isObject(obj)) {  
       throw new Error('obj 不是一个对象！')  
  }  
​  
   let isArray = Array.isArray(obj)  
   let cloneObj = isArray ? [] : {}  
   for (let key in obj) {  
       cloneObj[key] = isObject(obj[key]) ? deepClone(obj[key]) : obj[key]  
  }  
​  
   return cloneObj  
}

注意：for...in 法不支持拷贝 func、date、reg 和 err

// 代理法  
function deepClone(obj) {  
   if (!isObject(obj)) {  
       throw new Error('obj 不是一个对象！')  
  }  
​  
   let isArray = Array.isArray(obj)  
   let cloneObj = isArray ? [...obj] : { ...obj }  
   Reflect.ownKeys(cloneObj).forEach(key => {  
       cloneObj[key] = isObject(obj[key]) ? deepClone(obj[key]) : obj[key]  
  })  
​  
   return cloneObj  
}

## 说说严格模式的限制

* 严格模式主要有以下限制：
* 变量必须声明后再使用
* 函数的参数不能有同名属性，否则报错
* 不能使用 with 语句
* 不能对只读属性赋值，否则报错
* 不能使用前缀 0 表示八进制数，否则报错
* 不能删除不可删除的属性，否则报错
* 不能删除变量 delete prop，会报错，只能删除属性 delete global[prop]
* eval 不会在它的外层作用域引入变量
* eval 和 arguments 不能被重新赋值
* arguments 不会自动反映函数参数的变化
* 不能使用 arguments.callee
* 不能使用 arguments.caller
* 禁止 this 指向全局对象
* 不能使用 fn.caller 和 fn.arguments 获取函数调用的堆栈
* 增加了保留字（比如 protected、static 和 interface）

## 编写一个方法 求一个字符串的字节长度

假设：一个英文字符占用一个字节，一个中文字符占用两个字节

function GetBytes(str){​  
       var len = str.length;​  
       var bytes = len;​  
       for(var i=0; i<len; i++){​  
           if (str.charCodeAt(i) > 255) bytes++;​  
      }​  
       return bytes;​  
  }​  
alert(GetBytes("你好,as"));

## 请解释什么是事件代理

事件代理（Event Delegation），又称之为事件委托。是 JavaScript 中常用绑定事件的常用技巧。顾名思义，“事件代理”即是把原本需要绑定的事件委托给父元素，让父元素担当事件监听的职务。事件代理的原理是 DOM 元素的事件冒泡。使用事件代理的好处是可以提高性能

## attribute 和 property 的区别是什么？

* attribute 是 dom 元素在文档中作为 html 标签拥有的属性；
* property 就是 dom 元素在 js 中作为对象拥有的属性。
* 对于 html 的标准属性来说，attribute 和 property 是同步的，是会自动更新的
* 但是对于自定义的属性来说，他们是不同步的

## 页面编码和被请求的资源编码如果不一致如何处理？

* 后端响应头设置 charset
* 前端页面<meta>设置 charset

## 把 <script> 放在 </body> 之前和之后有什么区别？浏览器会如何解析它们？

按照 HTML 标准，在</body>结束后出现<script>或任何元素的开始标签，都是解析错误 虽然不符合 HTML 标准，但浏览器会自动容错，使实际效果与写在</body>之前没有区别 浏览器的容错机制会忽略<script>之前的</body>，视作<script>仍在 body 体内。省略</body>和</html>闭合标签符合 HTML 标准，服务器可以利用这一标准

## 把 <script> 放在 </head> 中会有什么问题？

在浏览器渲染页面之前，它需要通过解析HTML标记然后构建DOM树。在这个过程中，如果解析器遇到了一个脚本(script)，它就会停下来，并且执行这个脚本，然后才会继续解析HTML。如果遇到了一个引用外部资源的脚本(script)，它就必须停下来等待这个脚本资源的下载，而这个行为会导致一个或者多个的网络往返，并且会延迟页面的首次渲染时间。

还有一点是需要我们注意的，那就是外部引入的脚本(script)会阻塞浏览器的并行下载，HTTP/1.1规范表明，浏览器在每个主机下并行下载的组件不超过两个(也就是说，浏览器一次只能够同时从同一个服务器加载两个脚本)；如果你网站的图片是通过多个服务器提供的，那么按道理来说，你的网站可以一次并行下载多张图片。但是，当我们网站在加载脚本的时候；浏览器不会再启动任何其它的下载，即使这些组件来自不同的服务器。

## JavaScript 中，调用函数有哪几种方式？

* 方法调用模式 Foo.foo(arg1, arg2);
* 函数调用模式 foo(arg1, arg2);
* 构造器调用模式 (new Foo())(arg1, arg2);
* call/applay 调用模式 Foo.foo.call(that, arg1, arg2);
* bind 调用模式 Foo.foo.bind(that)(arg1, arg2)();

## 简单实现 Function.bind 函数？

 if (!Function.prototype.bind) {  
   Function.prototype.bind = function(that) {  
     var func = this, args = arguments;  
     return function() {  
       return func.apply(that, Array.prototype.slice.call(args, 1));  
    }  
  }  
}  
 // 只支持 bind 阶段的默认参数：  
 func.bind(that, arg1, arg2)();  
​  
 // 不支持以下调用阶段传入的参数：  
 func.bind(that)(arg1, arg2);

## 列举一下 JavaScript 数组和对象有哪些原生方法

* 数组：
  + arr.concat(arr1, arr2, arrn);
  + arr.join(",");
  + arr.sort(func);
  + arr.pop();
  + arr.push(e1, e2, en);
  + arr.shift();
  + unshift(e1, e2, en);
  + arr.reverse();
  + arr.slice(start, end);
  + arr.splice(index, count, e1, e2, en);
  + arr.indexOf(el);
  + arr.includes(el); // ES6
* 对象：
  + object.hasOwnProperty(prop);
  + object.propertyIsEnumerable(prop);
  + object.valueOf();
  + object.toString();
  + object.toLocaleString();
  + Class.prototype.isPropertyOf(object);

## Array.slice() 与 Array.splice() 的区别？

* slice -- “读取”数组指定的元素，不会对原数组进行修改
  + 语法：arr.slice(start, end)
  + start 指定选取开始位置（含）
  + end 指定选取结束位置（不含）
* splice
  + “操作”数组指定的元素，会修改原数组，返回被删除的元素
  + 语法：arr.splice(index, count, [insert Elements])
  + index 是操作的起始位置
  + count = 0 插入元素，count > 0 删除元素
  + [insert Elements] 向数组新插入的元素

## 在 javascript 中，1 与 Number(1)有什么区别 [易混淆]

var a = Number(1) // 1  
var b = new Number(1)  // Number {[[PrimitiveValue]]: 1}  
typeof (a) // number  
typeof (b) // object  
a == b // true

* var a = 1 是一个常量，而 Number(1)是一个函数
* new Number(1)返回的是一个对象
* a==b 为 true 是因为所以在求值过程中，总是会强制转为原始数据类型而非对象，例如下面的代码:

typeof 123 // "number"  
typeof new Number(123) // "object"  
123 instanceof Number // false  
(new Number(123)) instanceof Number // true  
123 === new Number(123) // false

## console.log(!!(new Boolean(false))输出什么 [易混淆]

true

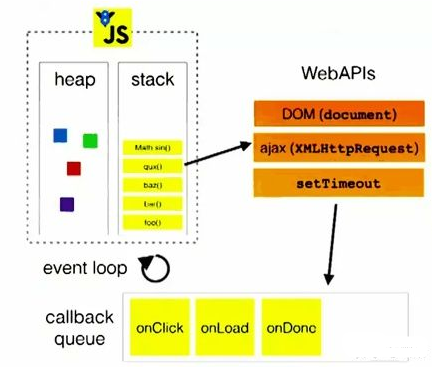
布尔的包装对象 Boolean 的对象实例，对象只有在 null 与 undefined 时，才会认定为布尔的 false 值，布尔包装对象本身是个对象，对象->布尔 都是 true，所以 new Boolean(false)其实是布尔的 true，看下面这段代码:

if(new Boolean(false)){  
   alert('true!!');  
}

只有使用了 valueOf 后才是真正的转换布尔值，与上面包装对象与原始资料转换说明的相同:

!!(new Boolean(false))  //true  
(new Boolean(false)).valueOf() //false

## 浏览器中的 Event Loop



* 主线程运行的时候会生成堆（heap）和栈（stack）；
* js 从上到下解析方法，将其中的同步任务按照执行顺序排列到执行栈中；
* 当程序调用外部的 API 时，比如 ajax、setTimeout 等，会将此类异步任务挂起，继续执行执行栈中的任务，等异步任务返回结果后，再按照执行顺序排列到事件队列中；
* 主线程先将执行栈中的同步任务清空，然后检查事件队列中是否有任务，如果有，就将第一个事件对应的回调推到执行栈中执行，若在执行过程中遇到异步任务，则继续将这个异步任务排列到事件队列中。
* 主线程每次将执行栈清空后，就去事件队列中检查是否有任务，如果有，就每次取出一个推到执行栈中执行，这个过程是循环往复的... ...，这个过程被称为“Event Loop 事件循环”

## 正则表达式

**1、写一个function，清除字符串前后的空格。（兼容所有浏览器）**

function trim(str) {

if (str &&typeof str === "string") {

return str.replace(/(^\s\*)|(\s\*)$/g,""); //去除前后空白符

}

}

**2、使用正则表达式验证邮箱格式**

var reg = /^(\w)+(\.\w+)\*@(\w)+((\.\w{2,3}){1,3})$/;

var email = "example@qq.com";

console.log(reg.test(email)); // true

# JavaScript高级

## 请指出document load和document ready的区别？

共同点：这两种事件都代表的是页面文档加载时触发。

异同点：

ready 事件的触发，表示文档结构已经加载完成（不包含图片等非文字媒体文件）。

onload 事件的触发，表示页面包含图片等文件在内的所有元素都加载完成。

## 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

## 什么是 Window 对象? 什么是 Document 对象?

* Window 对象表示当前浏览器的窗口，是 JavaScript 的顶级对象。
* 我们创建的所有对象、函数、变量都是 Window 对象的成员。
* Window 对象的方法和属性是在全局范围内有效的。
* Document 对象是 HTML 文档的根节点与所有其他节点（元素节点，文本节点，属性节点, 注释节点）
* Document 对象使我们可以通过脚本对 HTML 页面中的所有元素进行访问
* Document 对象是 Window 对象的一部分，可通过 window.document 属性对其进行访问

## 介绍 DOM 的发展

* DOM：文档对象模型（Document Object Model），定义了访问 HTML 和 XML 文档的标准，与编程语言及平台无关
* DOM0：提供了查询和操作 Web 文档的内容 API。未形成标准，实现混乱。如：document.forms['login']
* DOM1：W3C 提出标准化的 DOM，简化了对文档中任意部分的访问和操作。如：JavaScript 中的 Document 对象
* DOM2：原来 DOM 基础上扩充了鼠标事件等细分模块，增加了对 CSS 的支持。如：getComputedStyle(elem, pseudo)
* DOM3：增加了 XPath 模块和加载与保存（Load and Save）模块。如：XPathEvaluator

## 介绍 DOM0，DOM2，DOM3 事件处理方式区别

DOM0 级事件处理方式：

* btn.onclick = func;
* btn.onclick = null;

DOM2 级事件处理方式：

* btn.addEventListener('click', func, false);
* btn.removeEventListener('click', func, false);
* btn.attachEvent("onclick", func);
* btn.detachEvent("onclick", func);

DOM3 级事件处理方式：

* eventUtil.addListener(input, "textInput", func);
* eventUtil 是自定义对象，textInput 是 DOM3 级事件

## JavaScript 实现异步编程的方法？

* 回调函数
* 事件监听
* 发布/订阅
* Promises 对象
* Async 函数[ES7]

## DOM 操作——怎样添加、移除、移动、复制、创建和查找节点?

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

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

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

## 用过哪些设计模式？

1. 工厂模式：

* 主要好处就是可以消除对象间的耦合，通过使用工程方法而不是 new 关键字。将所有实例化的代码集中在一个位置防止代码重复
* 工厂模式解决了重复实例化的问题 ，但还有一个问题,那就是识别问题，因为根本无法 搞清楚他们到底是哪个对象的实例

function createObject(name,age,profession){  
   //集中实例化的函数  
   var obj = new Object();  
   obj.name = name;  
   obj.age = age;  
   obj.profession = profession;  
   obj.move = function () {  
       return this.name + ' at ' + this.age + ' engaged in ' + this.profession;  
  };  
   return obj;  
}  
var test1 = createObject('trigkit4',22,'programmer');//第一个实例var test2 = createObject('mike',25,'engineer');//第二个实例

1. 构造函数模式

* 使用构造函数的方法 ，即解决了重复实例化的问题 ，又解决了对象识别的问题，该模式与工厂模式的不同之处在于
* 构造函数方法没有显示的创建对象 (new Object());
* 直接将属性和方法赋值给 this 对象;
* 没有 renturn 语句

## 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）

## Javascript 垃圾回收方法

标记清除（mark and sweep）

* 这是 JavaScript 最常见的垃圾回收方式，当变量进入执行环境的时候，比如函数中声明一个变量，垃圾回收器将其标记为“进入环境”，当变量离开环境的时候（函数执行结束）将其标记为“离开环境”
* 垃圾回收器会在运行的时候给存储在内存中的所有变量加上标记，然后去掉环境中的变量以及被环境中变量所引用的变量（闭包），在这些完成之后仍存在标记的就是要删除的变量了

引用计数(reference counting)

* 在低版本 IE 中经常会出现内存泄露，很多时候就是因为其采用引用计数方式进行垃圾回收。引用计数的策略是跟踪记录每个值被使用的次数，当声明了一个 变量并将一个引用类型赋值给该变量的时候这个值的引用次数就加 1，如果该变量的值变成了另外一个，则这个值得引用次数减 1，当这个值的引用次数变为 0 的时 候，说明没有变量在使用，这个值没法被访问了，因此可以将其占用的空间回收，这样垃圾回收器会在运行的时候清理掉引用次数为 0 的值占用的空间

## 为什么 JS 是单线程,而不是多线程 [常考]

* 单线程是指 JavaScript 在执行的时候，有且只有一个主线程来处理所有的任务。
* 目的是为了实现与浏览器交互。
* 我们设想一下，如果 JavaScript 是多线程的，现在我们在浏览器中同时操作一个 DOM，一个线程要求浏览器在这个 DOM 中添加节点，而另一个线程却要求浏览器删掉这个 DOM 节点，那这个时候浏览器就会很郁闷，他不知道应该以哪个线程为准。所以为了避免此类现象的发生，降低复杂度，JavaScript 选择只用一个主线程来执行代码，以此来保证程序执行的一致性。

## JavaScript 对象生命周期的理解？

* 当创建一个对象时，JavaScript 会自动为该对象分配适当的内存
* 垃圾回收器定期扫描对象，并计算引用了该对象的其他对象的数量
* 如果被引用数量为 0，或惟一引用是循环的，那么该对象的内存即可回收

## 如何派发事件(dispatchEvent)？（如何进行事件广播？）

* W3C: 使用 dispatchEvent 方法
* IE: 使用 fireEvent 方法

var fireEvent = function(element, event){  
   if (document.createEventObject){  
       var mockEvent = document.createEventObject();  
       return element.fireEvent('on' + event, mockEvent)  
  }else{  
       var mockEvent = document.createEvent('HTMLEvents');  
       mockEvent.initEvent(event, true, true);  
       return !element.dispatchEvent(mockEvent);  
  }  
}

## 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
* 如果当前作用域没有找到属性或方法，会向上层作用域查找，直至全局函数，这种形式就是作用域链

谈谈 this 对象的理解

* this 总是指向函数的直接调用者
* 如果有 new 关键字，this 指向 new 出来的实例对象
* 在事件中，this 指向触发这个事件的对象

IE 下 attachEvent 中的 this 总是指向全局对象 Window

# 数据结构

## 什么是数据结构？

数据结构是计算机存储、组织数据的方式。对于特定的数据结构(比如数组)，有些操作效率很高(读某个数组元素)，有些操作的效率很低(删除某个数组元素)。程序员的目标是为当前的问题选择最优的数据结构。

## 8种常用数据结构

数组、栈、队列、链表、图、树、前缀树、哈希表

## 撤回，即Ctrl+Z，你知道它是怎么实现的吗？

把之前的应用状态(限制个数)保存到内存中，最近的状态放到第一个。这时，我们需要栈(stack)来实现这个功能。

栈中的元素采用LIFO (Last In First Out)，即后进先出。

# jQuery

## 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

* 创建 XMLHttpRequest 对象,也就是创建一个异步调用对象
* 建一个新的 HTTP 请求,并指定该 HTTP 请求的方法、URL 及验证信息
* 设置响应 HTTP 请求状态变化的函数
* 发送 HTTP 请求
* 获取异步调用返回的数据
* 用 JavaScript 和 DOM 实现局部刷新

## Ajax的优缺点及工作原理？

**定义和用法:**

AJAX = Asynchronous JavaScript and XML（异步的 JavaScript 和 XML）。Ajax 是一种用于创建快速动态网页的技术。Ajax 是一种在无需重新加载整个网页的情况下，能够更新部分网页的技术。

传统的网页（不使用 Ajax）如果需要更新内容，必须重载整个网页页面。

**优点：**

1.减轻服务器的负担,按需取数据,最大程度的减少冗余请求

2.局部刷新页面,减少用户心理和实际的等待时间,带来更好的用户体验

3.基于xml标准化,并被广泛支持,不需安装插件等,进一步促进页面和数据的分离

**缺点：**

1.AJAX大量的使用了javascript和ajax引擎,这些取决于浏览器的支持.在编写的时候考虑对浏览器的兼容性.

2.AJAX只是局部刷新,所以页面的后退按钮是没有用的.

3.对流媒体还有移动设备的支持不是太好等

**AJAX的工作原理**

1.创建ajax对象（XMLHttpRequest/ActiveXObject(Microsoft.XMLHttp)）

2.判断数据传输方式(GET/POST)

3.打开链接 open()

4.发送 send()

5.当ajax对象完成第四步（onreadystatechange）数据接收完成，判断http响应状态（status）200-300之间或者304（缓存）执行回调函数

## jQuery 库中的 $() 是什么？

　　$() 函数是 jQuery() 函数的别称。$() 函数用于将任何对象包裹成 jQuery 对象，接着你就被允许调用定义在 jQuery 对象上的多个不同方法。你可以将一个选择器字符串传入 $() 函数，它会返回一个包含所有匹配的 DOM 元素数组的 jQuery 对象。

## 如何找到所有 HTML select 标签的选中项？

$('[name=selectname] :selected')

## $(this) 和 this 关键字在 jQuery 中有何不同？

$(this) 返回一个 jQuery 对象，你可以对它调用多个 jQuery 方法，比如用 text() 获取文本，用val() 获取值等等。

而 this 代表当前元素，它是 JavaScript 关键词中的一个，表示上下文中的当前 DOM 元素。你不能对它调用 jQuery 方法，直到它被 $() 函数包裹，例如 $(this)。

## jquery怎么移除标签onclick属性？

获得a标签的onclick属性: $("a").attr("onclick")

删除onclick属性：$("a").removeAttr("onclick");

设置onclick属性：$("a").attr("onclick","test();");

## jquery中addClass,removeClass,toggleClass的使用。

$(selector).addClass(class)：为每个匹配的元素添加指定的类名

$(selector).removeClass(class)：从所有匹配的元素中删除全部或者指定的类，删除class中某个值；

$(selector).toggleClass(class)：如果存在（不存在）就删除（添加）一个类

$(selector).removeAttr(class);删除class这个属性；

## JQuery有几种选择器?

(1)、基本选择器：#id，class,element,\*;

(2)、层次选择器：parent > child，prev + next ，prev ~ siblings

(3)、基本[过滤器](http://zhidao.baidu.com/search?word=%E8%BF%87%E6%BB%A4%E5%99%A8&fr=qb_search_exp&ie=utf8" \t "_blank)选择器：:first，:last ，:not ，:even ，:odd ，:eq ，:gt ，:lt

(4)、内容[过滤器](http://zhidao.baidu.com/search?word=%E8%BF%87%E6%BB%A4%E5%99%A8&fr=qb_search_exp&ie=utf8" \t "_blank)选择器： :contains ，:empty ，:has ，:parent

(5)、可见性[过滤器](http://zhidao.baidu.com/search?word=%E8%BF%87%E6%BB%A4%E5%99%A8&fr=qb_search_exp&ie=utf8" \t "_blank)选择器：:hidden ，:visible

(6)、属性过滤器选择器：[attribute] ，[attribute=value] ，[attribute!=value] ，[attribute^=value] ，[attribute$=value] ，[attribute\*=value]

(7)、子元素过滤器选择器：:nth-child ，:first-child ，:last-child ，:only-child

(8)、表单选择器： :input ，:text ，:password ，:radio ，:checkbox ，:submit 等；

(9)、表单过滤器选择器：:enabled ，:disabled ，:checked ，:selected

## jQuery中的Delegate()函数有什么作用？

   delegate()会在以下两个情况下使用到：

 1、如果你有一个父元素，需要给其下的子元素添加事件，这时你可以使用delegate()了，代码如下：

$("ul").delegate("li", "click", function(){ $(this).hide(); });

 2、当元素在当前页面中不可用时，可以使用delegate()

## $(document).ready()方法和window.onload有什么区别？

 (1)、window.onload方法是在网页中所有的元素(包括元素的所有关联文件)完全加载到浏览器后才执行的。

 (2)、$(document).ready() 方法可以在DOM载入就绪时就对其进行操纵，并调用执行绑定的函数。

## 如何用jQuery禁用浏览器的前进后退按钮？

实现代码如下：

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

　　$(document).ready(function() {

　　　　window.history.forward(1);

　　　　//OR window.history.forward(-1);

　　});

</script>

## jquery中$.get()提交和$.post()提交有区别吗？

相同点：都是异步请求的方式来获取服务端的数据；

异同点：

1、请求方式不同：$.get() 方法使用GET方法来进行异步请求的。$.post() 方法使用POST方法来进行异步请求的。

2、参数传递方式不同：get请求会将参数跟在URL后进行传递，而POST请求则是作为HTTP消息的实体内容发送给Web服务器的，这种传递是对用户不可见的。

3、数据传输大小不同：get方式传输的数据大小不能超过2KB 而POST要大的多

4、安全问题： GET 方式请求的数据会被浏览器缓存起来，因此有安全问题。

## 写出一个简单的$.ajax()的请求方式？

$.ajax({

url:'http://www.baidu.com',

type:'POST',

data:data,

cache:true,

headers:{},

beforeSend：function(){},

success:function(){},

error:function(){},

complete:function(){}

});

## jQuery的事件委托方法bind 、live、delegate、on之间有什么区别？

**(1)、bind 【jQuery 1.3之前】**

定义和用法：主要用于给选择到的元素上绑定特定事件类型的监听函数；

语法：bind(type,[data],function(eventObject))；

特点：

　　(1)、适用于页面元素静态绑定。只能给调用它的时候已经存在的元素绑定事件，不能给未来新增的元素绑定事件。

　　(2)、当页面加载完的时候，你才可以进行bind()，所以可能产生效率问题。

实例如下：$( "#members li a" ).bind( "click", function( e ) {} );

**(2)、live 【jQuery 1.3之后】**

定义和用法：主要用于给选择到的元素上绑定特定事件类型的监听函数；

语法：live(type, [data], fn);

特点：

　　(1)、live方法并没有将监听器绑定到自己(this)身上，而是绑定到了this.context上了。

　　(2)、live正是利用了事件委托机制来完成事件的监听处理，把节点的处理委托给了document，新添加的元素不必再绑定一次监听器。

　　(3)、使用live（）方法但却只能放在直接选择的元素后面，不能在层级比较深，连缀的DOM遍历方法后面使用，即$(“ul”").live...可以，但$("body").find("ul").live...不行；

实例如下：$( document ).on( "click", "#members li a", function( e ) {} );

**(3)、delegate 【jQuery 1.4.2中引入】**

定义和用法：将监听事件绑定在就近的父级元素上

语法：delegate(selector,type,[data],fn)

特点：

　　(1)、选择就近的父级元素，因为事件可以更快的冒泡上去，能够在第一时间进行处理。

　　(2)、更精确的小范围使用事件代理，性能优于.live()。可以用在动态添加的元素上。

实例如下：

$("#info\_table").delegate("td","click",function(){/\*显示更多信息\*/});

$("table").find("#info").delegate("td","click",function(){/\*显示更多信息\*/});

**(4)、on 【1.7版本整合了之前的三种方式的新事件绑定机制】**

定义和用法：将监听事件绑定到指定元素上。

语法：on(type,[selector],[data],fn)

实例如下：$("#info\_table").on("click","td",function(){/\*显示更多信息\*/});参数的位置写法与delegate不一样。

说明：on方法是当前JQuery推荐使用的事件绑定方法，附加只运行一次就删除函数的方法是one()。

 总结：.bind(), .live(), .delegate(),.on()分别对应的相反事件为：.unbind(),.die(), .undelegate(),.off()

# BootStrap

## 1.为什么使用bootstrap?

Bootstrap具有移动设备优先、浏览器支持良好、容易上手、响应式设计等优点，所以Bootstrap被广泛应用。

## 2.为了让 Bootstrap 开发的网站对移动设备友好，确保适当的绘制和触屏缩放，需要在网页的 head 之中添加 viewport meta 标签，width 属性控制设备的宽度。

假设您的网站将被带有不同屏幕分辨率的设备浏览，那么将它设置为 device-width 可以确保它能正确呈现在不同设备上。

initial-scale=1.0 确保网页加载时，以 1:1 的比例呈现，不会有任何的缩放。

在移动设备浏览器上，通过为 viewport meta 标签添加 user-scalable=no 可以禁用其缩放（zooming）功能。

通常情况下，maximum-scale=1.0 与 user-scalable=no 一起使用。这样禁用缩放功能后，用户只能滚动屏幕，就能让您的网站看上去更像原生应用的感觉，所以要加上以下代码：

<metaname="viewport" content="width=device-width,initial-scale=1">

## 3.对于各类尺寸的设备，Bootstrap设置的class前缀分别是什么？

超小设备手机（<768px）：.col-xs-

小型设备平板电脑（>=768px）：.col-sm-

中型设备台式电脑（>=992px）：.col-md-

大型设备台式电脑（>=1200px）：.col-lg-

## 4.Bootstrap如何设置响应式表格？

增加class="table-responsive"

## 5、使用Bootstrap创建垂直表单的基本步骤？

（1）向父<form>元素添加role="form"；

（2）把标签和控件放在一个带有class="form-group"的<div>中，这是获取最佳间距所必需的；

（3）向所有的文本元素<input>、<textarea>、<select>添加class="form-control"。

## 6、使用Bootstrap创建水平表单的基本步骤？

（1）向父<form>元素添加class="form-horizontal"；

（2）把标签和控件放在一个带有class="form-group"的<div>中；

（3）向标签添加class="control-label"。

## 7、使用Bootstrap如何创建表单控件的帮助文本？

增加class="help-block"的span标签或p标签。

## 8、使用Bootstrap激活或禁用按钮要如何操作？

激活按钮：给按钮增加.active的class

禁用按钮：给按钮增加disabled="disabled"的属性

## 9、Bootstrap有哪些关于<img>的class？

（1）.img-rounded 为图片添加圆角

（2）.img-circle 将图片变为圆形

（3）.img-thumbnail 缩略图功能

（4）.img-responsive 图片响应式 (将很好地扩展到父元素)

## 10、Bootstrap中有关元素浮动及清除浮动的class？

（1）class="pull-left" 元素浮动到左边

（2）class="pull-right" 元素浮动到右边

（3）class="clearfix" 清除浮动

## 11、除了屏幕阅读器外，其他设备上隐藏元素的class？

class="sr-only"

## 12、Bootstrap如何制作下拉菜单？

（1）将下拉菜单包裹在class="dropdown"的<div>中；

（2）在触发下拉菜单的按钮中添加：class="btn dropdown-toggle" id="dropdownMenu1" data-toggle="dropdown"

（3）在包裹下拉菜单的ul中添加：class="dropdown-menu" role="menu" aria-labelledby="dropdownMenu1"

（4）在下拉菜单的列表项中添加：role="presentation"。其中，下拉菜单的标题要添加class="dropdown-header"，选项部分要添加tabindex="-1"。

## 13、Bootstrap如何制作按钮组？以及水平按钮组和垂直按钮组的优先级？

（1）用class="btn-group"的<div>去包裹按钮组；class="btn-group-vertical"可设置垂直按钮组。

（2）btn-group的优先级高于btn-group-vertical的优先级。

## 14、Bootstrap如何设置按钮的下拉菜单？

在一个 .btn-group 中放置按钮和下拉菜单即可。

## 15、Bootstrap中的输入框组如何制作？

（1）把前缀或者后缀元素放在一个带有class="input-group"中的<div>中；

（2）在该<div>内，在class="input-group-addon"的<span>里面放置额外的内容；

（3）把<span>放在<input>元素的前面或后面。

## 16、Bootstrap中的导航都有哪些？

（1）导航元素：有class="nav nav-tabs"的标签页导航，还有class="nav nav-pills"的胶囊式标签页导航；

（2）导航栏：class="navbar navbar-default" role="navigation"；

（3）面包屑导航：class="breadcrumb"

## 17、Bootstrap中设置分页的class？

默认的分页：class="pagination"

默认的翻页：class="pager"

## 18、Bootstrap中显示标签的class？

class="label"

## 19、Bootstrap中如何制作徽章？

<span class="badge">26</span>

20、Bootstrap中超大屏幕的作用是什么？

设置class="jumbotron"可以制作超大屏幕，该组件可以增加标题的大小并增加更多的外边距。

# NodeJS

## 为什么用Nodejs，它有哪些缺点？

 答：事件驱动，通过闭包很容易实现客户端的生命活期。  
    不用担心多线程，锁，并行计算的问题  
    V8引擎速度非常快  
    对于游戏来说，写一遍游戏逻辑代码，前端后端通用。  
当然Nodejs也有一些缺点：

    nodejs更新很快，可能会出现版本联兼容  
    nodejs还不算成熟，还没有大制作。  
    nodejs不像其他的服务器，对于不同的连接，不支持进程和线程操作。

## sessionStorage 、localStorage 和 cookie 之间的区别

 共同点：用于浏览器端存储的缓存数据

不同点：

(1)、存储内容是否发送到服务器端：当设置了Cookie后，数据会发送到服务器端，造成一定的宽带浪费；

        web storage,会将数据保存到本地，不会造成宽带浪费；

(2)、数据存储大小不同：Cookie数据不能超过4K,适用于会话标识；web storage数据存储可以达到5M;

(3)、数据存储的有效期限不同：cookie只在设置了Cookid过期时间之前一直有效，即使关闭窗口或者浏览器；

        sessionStorage,仅在关闭浏览器之前有效；localStorage,数据存储永久有效；

(4)、作用域不同：cookie和localStorage是在同源同窗口中都是共享的；sessionStorage不在不同的浏览器窗口中共享，即使是同一个页面；

## Web Storage与Cookie相比存在的优势：

(1)、存储空间更大：IE8下每个独立的存储空间为10M，其他浏览器实现略有不同，但都比Cookie要大很多。

(2)、存储内容不会发送到服务器：当设置了Cookie后，Cookie的内容会随着请求一并发送的服务器，这对于本地存储的数据是一种带宽浪费。而Web Storage中的数据则仅仅是存在本地，不会与服务器发生任何交互。

(3)、更多丰富易用的接口：Web Storage提供了一套更为丰富的接口，如setItem,getItem,removeItem,clear等,使得数据操作更为简便。cookie需要自己封装。

(4)、独立的存储空间：每个域（包括子域）有独立的存储空间，各个存储空间是完全独立的，因此不会造成数据混乱。

## web 开发中会话跟踪的方法有哪些

* cookie
* session
* url 重写
* 隐藏 input
* ip 地址

## 如何删除一个 cookie

将时间设为当前时间往前一点

var date = new Date();  
date.setDate(date.getDate() - 1);//真正的删除

setDate()方法用于设置一个月的某一天

expires 的设置

 document.cookie = 'user='+ encodeURIComponent('name')  + ';expires = ' + new Date(0)

## 什么是错误优先的回调函数？

错误优先的回调函数用于传递错误和数据。第一个参数始终应该是一个错误对象， 用于检查程序是否发生了错误。其余的参数用于传递数据。

解析：这个题目的主要作用在于检查被面试者对于Node中异步操作的一些基本知识的掌握。

## 如何避免回调地狱

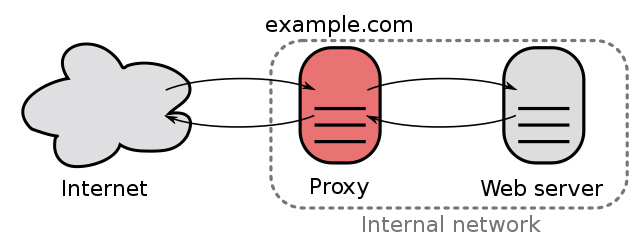
你可以有如下几个方法：

* 模块化：将回调函数分割为独立的函数
* 使用Promises
* 使用yield来计算生成器或Promise

解析：这个问题有很多种答案，取决你使用的场景，例如ES6, ES7，或者一些控制流库。

## 如何用Node监听80端口

这题有陷阱！在类Unix系统中你不应该尝试去监听80端口，因为这需要超级用户权限。 因此不推荐让你的应用直接监听这个端口。

目前，如果你一定要让你的应用监听80端口的话，你可以有通过在Node应用的前方再增加一层反向代理 （例如[nginx](http://nginx.org/)）来实现，如下图所示。否则，建议你直接监听大于1024的端口。

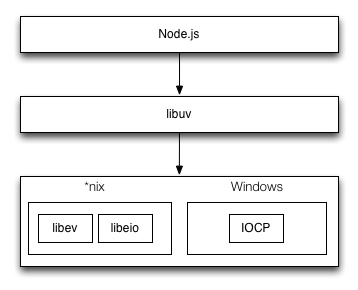
方向代理指的是以代理服务器来接收Internet上的连接请求，然后将请求转发给内部网络上的服务器， 并且将服务器返回的结果发送给客户端。

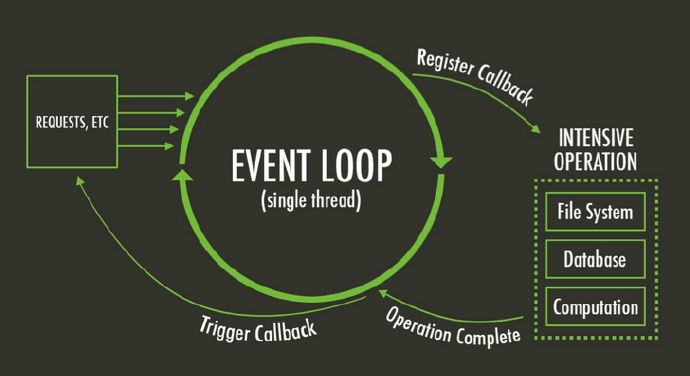
关于反向代理的更多内容，建议你阅读[这篇文章](http://www.cnblogs.com/edisonchou/p/4126742.html)。

解释：这个问题用于检查被面试者是否有实际运行Node应用的经验。

## 什么是事件循环

Node采用的是单线程的处理机制（所有的I/O请求都采用非阻塞的工作方式），至少从Node.js开发者的角度是这样的。 而在底层，Node.js借助[libuv](https://github.com/libuv/libuv)来作为抽象封装层， 从而屏蔽不同操作系统的差异，Node可以借助livuv来来实现多线程。下图表示了Node和libuv的关系。



Libuv库负责Node API的执行。它将不同的任务分配给不同的线程，形成一个事件循环， 以异步的方式将任务的执行结果返回给V8引擎。可以简单用下面这张图来表示。

每一个I/O都需要一个回调函数——一旦执行完便推到事件循环上用于执行。

解释：这用于检查Node.js的底层知识，例如什么是libuv，它的作用是什么。

## 使用NPM有哪些好处？

通过NPM，你可以安装和管理项目的依赖，并且能够指明依赖项的具体版本号。 对于Node应用开发而言，你可以通过package.json文件来管理项目信息，配置脚本， 以及指明项目依赖的具体版本。

关于NPM的更多信息，你可以参考[官方文档](https://docs.npmjs.com/files/package.json)。

解析：它能考察面试者使用npm命令的基础知识和Node.js开发的实际经验。

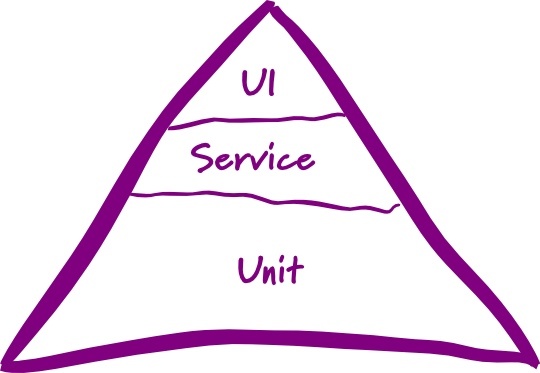
## 什么是Stub？举个使用场景

Stub是用于模拟一个组件或模块的函数或程序。在测试用例中， 简单的说，你可以用Stub去模拟一个方法，从而避免调用真实的方法， 使用Stub你还可以返回虚构的结果。你可以配合断言使用Stub。

在单元测试中：Stub是完全模拟一个外部依赖，而Mock常用来判断测试通过还是失败。

解析：用于测试被面试者是否有测试的经验。如果被面试者知道什么是Stub， 那么可以继续问他是如何做单元测试的。

## 什么是测试金字塔？

[测试金字塔](http://zyzhang.github.io/blog/2013/04/28/test-pyramid/)指的是： 当我们在编写测试用例时，底层的单元测试应该远比上层的端到端测试要多。

当我们谈到HTTP API时，我们可能会涉及到：

* 有很多针对模型的底层单元测试
* 但你需要测试模型间如何交互时，需要减少集成测试

解析：本文主要考察被面试者的在测试方面的经验。

# React

## React 中 keys 的作用是什么？

Keys 是 React 用于追踪哪些列表中元素被修改、被添加或者被移除的辅助标识。

render () {

return (

<ul>

{this.state.todoItems.map(({item, key}) => {

return <li key={key}>{item}</li>

})}

</ul>

)

}

在开发过程中，我们需要保证某个元素的 key 在其同级元素中具有唯一性。在 React Diff 算法中 React 会借助元素的 Key 值来判断该元素是新近创建的还是被移动而来的元素，从而减少不必要的元素重渲染。此外，React 还需要借助 Key 值来判断元素与本地状态的关联关系，因此我们绝不可忽视转换函数中 Key 的重要性。

## 调用 setState 之后发生了什么？

在代码中调用 setState 函数之后，React 会将传入的参数对象与组件当前的状态合并，然后触发所谓的调和过程（Reconciliation）。经过调和过程，React 会以相对高效的方式根据新的状态构建 React 元素树并且着手重新渲染整个 UI 界面。在 React 得到元素树之后，React 会自动计算出新的树与老树的节点差异，然后根据差异对界面进行最小化重渲染。在差异计算算法中，React 能够相对精确地知道哪些位置发生了改变以及应该如何改变，这就保证了按需更新，而不是全部重新渲染。

## react 生命周期函数

* 初始化阶段：
  + getDefaultProps:获取实例的默认属性
  + getInitialState:获取每个实例的初始化状态
  + componentWillMount：组件即将被装载、渲染到页面上
  + render:组件在这里生成虚拟的 DOM 节点
  + componentDidMount:组件真正在被装载之后
* 运行中状态：
  + componentWillReceiveProps:组件将要接收到属性的时候调用
  + shouldComponentUpdate:组件接受到新属性或者新状态的时候（可以返回 false，接收数据后不更新，阻止 render 调用，后面的函数不会被继续执行了）
  + componentWillUpdate:组件即将更新不能修改属性和状态
  + render:组件重新描绘
  + componentDidUpdate:组件已经更新
* 销毁阶段：
  + componentWillUnmount:组件即将销毁

## shouldComponentUpdate 是做什么的，（react 性能优化是哪个周期函数？）

shouldComponentUpdate 这个方法用来判断是否需要调用 render 方法重新描绘 dom。因为 dom 的描绘非常消耗性能，如果我们能在 shouldComponentUpdate 方法中能够写出更优化的 dom diff 算法，可以极大的提高性能。

参考[react 性能优化-sf](https://github.com/FEGuideTeam/FEGuide/blob/master/%E6%A1%86%E6%9E%B6/https/segmentfault.com/a/1190000006254212)

## 为什么虚拟 dom 会提高性能?(必考)

虚拟 dom 相当于在 js 和真实 dom 中间加了一个缓存，利用 dom diff 算法避免了没有必要的 dom 操作，从而提高性能。

用 JavaScript 对象结构表示 DOM 树的结构；然后用这个树构建一个真正的 DOM 树，插到文档当中当状态变更的时候，重新构造一棵新的对象树。然后用新的树和旧的树进行比较，记录两棵树差异把 2 所记录的差异应用到步骤 1 所构建的真正的 DOM 树上，视图就更新了。

## react diff 原理（常考，大厂必考）

* 把树形结构按照层级分解，只比较同级元素。
* 给列表结构的每个单元添加唯一的 key 属性，方便比较。
* React 只会匹配相同 class 的 component（这里面的 class 指的是组件的名字）
* 合并操作，调用 component 的 setState 方法的时候, React 将其标记为 dirty.到每一个事件循环结束, React 检查所有标记 dirty 的 component 重新绘制.
* 选择性子树渲染。开发人员可以重写 shouldComponentUpdate 提高 diff 的性能。

参考：[React 的 diff 算法](https://github.com/FEGuideTeam/FEGuide/blob/master/%E6%A1%86%E6%9E%B6/https/segmentfault.com/a/1190000000606216)

## React 中 refs 的作用是什么？

Refs 是 React 提供给我们的安全访问 DOM 元素或者某个组件实例的句柄。我们可以为元素添加 ref 属性然后在回调函数中接受该元素在 DOM 树中的句柄，该值会作为回调函数的第一个参数返回：

class CustomForm extends Component {

handleSubmit = () => {

console.log("Input Value: ", this.input.value)

}

render () {

return (

<form onSubmit={this.handleSubmit}>

<input

type='text'

ref={(input) => this.input = input} />

<button type='submit'>Submit</button>

</form>

)

}

}

上述代码中的 input 域包含了一个 ref 属性，该属性声明的回调函数会接收 input 对应的 DOM 元素，我们将其绑定到 this 指针以便在其他的类函数中使用。另外值得一提的是，refs 并不是类组件的专属，函数式组件同样能够利用闭包暂存其值：

function CustomForm ({handleSubmit}) {

let inputElement

return (

<form onSubmit={() => handleSubmit(inputElement.value)}>

<input

type='text'

ref={(input) => inputElement = input} />

<button type='submit'>Submit</button>

</form>

)

}

## 如果你创建了类似于下面的 Twitter 元素，那么它相关的类定义是啥样子的？

<Twitter username='tylermcginnis33'>

{(user) => user === null

? <Loading />

: <Badge info={user} />}

</Twitter>

import React, { Component, PropTypes } from 'react'

import fetchUser from 'twitter'

// fetchUser take in a username returns a promise

// which will resolve with that username's data.

class Twitter extends Component {

// finish this

}

如果你还不熟悉回调渲染模式（Render Callback Pattern），这个代码可能看起来有点怪。这种模式中，组件会接收某个函数作为其子组件，然后在渲染函数中以 props.children 进行调用：

import React, { Component, PropTypes } from 'react'

import fetchUser from 'twitter'

class Twitter extends Component {

state = {

user: null,

}

static propTypes = {

username: PropTypes.string.isRequired,

}

componentDidMount () {

fetchUser(this.props.username)

.then((user) => this.setState({user}))

}

render () {

return this.props.children(this.state.user)

}

}

这种模式的优势在于将父组件与子组件解耦和，父组件可以直接访问子组件的内部状态而不需要再通过 Props 传递，这样父组件能够更为方便地控制子组件展示的 UI 界面。譬如产品经理让我们将原本展示的 Badge 替换为 Profile，我们可以轻易地修改下回调函数即可：

<Twitter username='tylermcginnis33'>

{(user) => user === null

? <Loading />

: <Profile info={user} />}

</Twitter>

## 展示组件(Presentational component)和容器组件(Container component)之间有何不同

* 展示组件关心组件看起来是什么。展示专门通过 props 接受数据和回调，并且几乎不会有自身的状态，但当展示组件拥有自身的状态时，通常也只关心 UI 状态而不是数据的状态。
* 容器组件则更关心组件是如何运作的。容器组件会为展示组件或者其它容器组件提供数据和行为(behavior)，它们会调用 Flux actions，并将其作为回调提供给展示组件。容器组件经常是有状态的，因为它们是(其它组件的)数据源。

## 类组件(Class component)和函数式组件(Functional component)之间有何不同

* 类组件不仅允许你使用更多额外的功能，如组件自身的状态和生命周期钩子，也能使组件直接访问 store 并维持状态
* 当组件仅是接收 props，并将组件自身渲染到页面时，该组件就是一个 '无状态组件(stateless component)'，可以使用一个纯函数来创建这样的组件。这种组件也被称为哑组件(dumb components)或展示组件

## (组件的)状态(state)和属性(props)之间有何不同

* State 是一种数据结构，用于组件挂载时所需数据的默认值。State 可能会随着时间的推移而发生突变，但多数时候是作为用户事件行为的结果。
* Props(properties 的简写)则是组件的配置。props 由父组件传递给子组件，并且就子组件而言，props 是不可变的(immutable)。组件不能改变自身的 props，但是可以把其子组件的 props 放在一起(统一管理)。Props 也不仅仅是数据--回调函数也可以通过 props 传递。

## 何为受控组件(controlled component)

在 HTML 中，类似 <input>, <textarea> 和 <select> 这样的表单元素会维护自身的状态，并基于用户的输入来更新。当用户提交表单时，前面提到的元素的值将随表单一起被发送。但在 React 中会有些不同，包含表单元素的组件将会在 state 中追踪输入的值，并且每次调用回调函数时，如 onChange 会更新 state，重新渲染组件。一个输入表单元素，它的值通过 React 的这种方式来控制，这样的元素就被称为"受控元素"。

## 何为高阶组件(higher order component)

高阶组件是一个以组件为参数并返回一个新组件的函数。HOC 运行你重用代码、逻辑和引导抽象。最常见的可能是 Redux 的 connect 函数。除了简单分享工具库和简单的组合，HOC 最好的方式是共享 React 组件之间的行为。如果你发现你在不同的地方写了大量代码来做同一件事时，就应该考虑将代码重构为可重用的 HOC。

## 为什么建议传递给 setState 的参数是一个 callback 而不是一个对象

因为 this.props 和 this.state 的更新可能是异步的，不能依赖它们的值去计算下一个 state。

## 除了在构造函数中绑定 this，还有其它方式吗

你可以使用属性初始值设定项(property initializers)来正确绑定回调，create-react-app 也是默认支持的。在回调中你可以使用箭头函数，但问题是每次组件渲染时都会创建一个新的回调。

## (在构造函数中)调用 super(props) 的目的是什么

在 super() 被调用之前，子类是不能使用 this 的，在 ES2015 中，子类必须在 constructor 中调用 super()。传递 props 给 super() 的原因则是便于(在子类中)能在 constructor 访问 this.props。

## 应该在 React 组件的何处发起 Ajax 请求

在 React 组件中，应该在 componentDidMount 中发起网络请求。这个方法会在组件第一次“挂载”(被添加到 DOM)时执行，在组件的生命周期中仅会执行一次。更重要的是，你不能保证在组件挂载之前 Ajax 请求已经完成，如果是这样，也就意味着你将尝试在一个未挂载的组件上调用 setState，这将不起作用。在 componentDidMount 中发起网络请求将保证这有一个组件可以更新了。

## 描述事件在 React 中的处理方式。

为了解决跨浏览器兼容性问题，您的 React 中的事件处理程序将传递 SyntheticEvent 的实例，它是 React 的浏览器本机事件的跨浏览器包装器。

这些 SyntheticEvent 与您习惯的原生事件具有相同的接口，除了它们在所有浏览器中都兼容。有趣的是，React 实际上并没有将事件附加到子节点本身。React 将使用单个事件监听器监听顶层的所有事件。这对于性能是有好处的，这也意味着在更新 DOM 时，React 不需要担心跟踪事件监听器。

## createElement 和 cloneElement 有什么区别？

React.createElement():JSX 语法就是用 React.createElement()来构建 React 元素的。它接受三个参数，第一个参数可以是一个标签名。如 div、span，或者 React 组件。第二个参数为传入的属性。第三个以及之后的参数，皆作为组件的子组件。

React.createElement(

type,

[props],

[...children]

)

React.cloneElement()与 React.createElement()相似，不同的是它传入的第一个参数是一个 React 元素，而不是标签名或组件。新添加的属性会并入原有的属性，传入到返回的新元素中，而就的子元素奖杯替换。

React.cloneElement(

element,

[props],

[...children]

)

## React 中有三种构建组件的方式

React.createClass()、ES6 class 和无状态函数。

## react 组件的划分业务组件技术组件？

* 根据组件的职责通常把组件分为 UI 组件和容器组件。
* UI 组件负责 UI 的呈现，容器组件负责管理数据和逻辑。
* 两者通过 React-Redux 提供 connect 方法联系起来。

## 简述 flux 思想

Flux 的最大特点，就是数据的"单向流动"。

1. 用户访问 View
2. View 发出用户的 Action
3. Dispatcher 收到 Action，要求 Store 进行相应的更新
4. Store 更新后，发出一个"change"事件
5. View 收到"change"事件后，更新页面

## React 项目用过什么脚手架（本题是开放性题目）

creat-react-app Yeoman 等

## 了解 redux 么，说一下 redux 吧

* redux 是一个应用数据流框架，主要是解决了组件间状态共享的问题，原理是集中式管理，主要有三个核心方法，action，store，reducer，工作流程是 view 调用 store 的 dispatch 接收 action 传入 store，reducer 进行 state 操作，view 通过 store 提供的 getState 获取最新的数据，flux 也是用来进行数据操作的，有四个组成部分 action，dispatch，view，store，工作流程是 view 发出一个 action，派发器接收 action，让 store 进行数据更新，更新完成以后 store 发出 change，view 接受 change 更新视图。Redux 和 Flux 很像。主要区别在于 Flux 有多个可以改变应用状态的 store，在 Flux 中 dispatcher 被用来传递数据到注册的回调事件，但是在 redux 中只能定义一个可更新状态的 store，redux 把 store 和 Dispatcher 合并,结构更加简单清晰
* 新增 state,对状态的管理更加明确，通过 redux，流程更加规范了，减少手动编码量，提高了编码效率，同时缺点时当数据更新时有时候组件不需要，但是也要重新绘制，有些影响效率。一般情况下，我们在构建多交互，多数据流的复杂项目应用时才会使用它们

## redux 有什么缺点

* 一个组件所需要的数据，必须由父组件传过来，而不能像 flux 中直接从 store 取。
* 当一个组件相关数据更新时，即使父组件不需要用到这个组件，父组件还是会重新 render，可能会有效率影响，或者需要写复杂的 shouldComponentUpdate 进行判断。

# Vue

## 1、active-class是哪个组件的属性？嵌套路由怎么定义？

答：vue-router模块的router-link组件。

## 2、怎么定义vue-router的动态路由？怎么获取传过来的动态参数？

答：在router目录下的index.js文件中，对path属性加上/:id。  使用router对象的params.id

## 3、vue-router有哪几种导航钩子？

答：三种，一种是全局导航钩子：router.beforeEach(to,from,next)，作用：跳转前进行判断拦截。第二种：组件内的钩子；第三种：单独路由独享组件

## 4、scss是什么？在vue.cli中的安装使用步骤是？有哪几大特性？

答：css的预编译。

使用步骤：

第一步：用npm 下三个loader（sass-loader、css-loader、node-sass）

第二步：在build目录找到webpack.base.config.js，在那个extends属性中加一个拓展.scss

第三步：还是在同一个文件，配置一个module属性

第四步：然后在组件的style标签加上lang属性 ，例如：lang=”scss”

有哪几大特性:

1、可以用变量，例如（$变量名称=值）；

2、可以用混合器，例如（）

3、可以嵌套

## 5、mint-ui是什么？怎么使用？说出至少三个组件使用方法？

答：基于vue的前端组件库。npm安装，然后import样式和js，vue.use（mintUi）全局引入。在单个组件局部引入：import {Toast} from ‘mint-ui’。组件一：Toast(‘登录成功’)；组件二：mint-header；组件三：mint-swiper

## 6、v-model是什么？怎么使用？ vue中标签怎么绑定事件？

答：可以实现双向绑定，指令（v-class、v-for、v-if、v-show、v-on）。vue的model层的data属性。绑定事件：<input @click=doLog() />

## 7、axios是什么？怎么使用？描述使用它实现登录功能的流程？

答：请求后台资源的模块。npm install axios -S装好，然后发送的是跨域，需在配置文件中config/index.js进行设置。后台如果是Tp5则定义一个资源路由。js中使用import进来，然后.get或.post。返回在.then函数中如果成功，失败则是在.catch函数中

## 8、axios+tp5进阶中，调用axios.post(‘api/user’)是进行的什么操作？axios.put(‘api/user/8′)呢？

答：跨域，添加用户操作，更新操作。

## 9、什么是RESTful API？怎么使用?

答：是一个api的标准，无状态请求。请求的路由地址是固定的，如果是tp5则先路由配置中把资源路由配置好。标准有：.post .put .delete

## 10、vuex是什么？怎么使用？哪种功能场景使用它？

答：vue框架中状态管理。在main.js引入store，注入。新建了一个目录store，….. export 。场景有：单页应用中，组件之间的状态。音乐播放、登录状态、加入购物车

## 11、mvvm框架是什么？它和其它框架（jquery）的区别是什么？哪些场景适合？

答：一个model+view+viewModel框架，数据模型model，viewModel连接两个

区别：vue数据驱动，通过数据来显示视图层而不是节点操作。

场景：数据操作比较多的场景，更加便捷

## 12、自定义指令（v-check、v-focus）的方法有哪些？它有哪些钩子函数？还有哪些钩子函数参数？

答：全局定义指令：在vue对象的directive方法里面有两个参数，一个是指令名称，另外一个是函数。组件内定义指令：directives

钩子函数：bind（绑定事件触发）、inserted(节点插入的时候触发)、update（组件内相关更新）

钩子函数参数：el、binding

## 13、说出至少4种vue当中的指令和它的用法？

答：v-if：判断是否隐藏；v-for：数据循环出来；v-bind:class：绑定一个属性；v-model：实现双向绑定

## 14、vue-router是什么？它有哪些组件？

答：vue用来写路由一个插件。router-link、router-view

## 15、导航钩子有哪些？它们有哪些参数？

答：导航钩子有：a/全局钩子和组件内独享的钩子。b/beforeRouteEnter、afterEnter、beforeRouterUpdate、beforeRouteLeave

参数：有to（去的那个路由）、from（离开的路由）、next（一定要用这个函数才能去到下一个路由，如果不用就拦截）最常用就这几种

## 16、Vue的双向数据绑定原理是什么？

答：vue.js 是采用数据劫持结合发布者-订阅者模式的方式，通过Object.defineProperty()来劫持各个属性的setter，getter，在数据变动时发布消息给订阅者，触发相应的监听回调。

**具体步骤：**

**第一步：需要observe的数据对象进行递归遍历**，包括子属性对象的属性，都加上 setter和getter  
这样的话，给这个对象的某个值赋值，就会触发setter，那么就能监听到了数据变化

**第二步：compile解析模板指令**，将模板中的变量替换成数据，然后初始化渲染页面视图，并将每个指令对应的节点绑定更新函数，添加监听数据的订阅者，一旦数据有变动，收到通知，更新视图

**第三步：Watcher订阅者是Observer和Compile之间通信的桥梁**，主要做的事情是:  
1、在自身实例化时往属性订阅器(dep)里面添加自己  
2、自身必须有一个update()方法  
3、待属性变动dep.notice()通知时，能调用自身的update()方法，并触发Compile中绑定的回调，则功成身退。

**第四步：MVVM作为数据绑定的入口，整合Observer、Compile和Watcher三者**，通过Observer来监听自己的model数据变化，通过Compile来解析编译模板指令，最终利用Watcher搭起Observer和Compile之间的通信桥梁，达到数据变化 -> 视图更新；视图交互变化(input) -> 数据model变更的双向绑定效果。

ps：16题答案同样适合”**vue data是怎么实现的？”**此面试题**。**

## 17、请详细说下你对vue生命周期的理解？

答：总共分为8个阶段创建前/后，载入前/后，更新前/后，销毁前/后。

**创建前/后：** 在beforeCreated阶段，vue实例的挂载元素$el和数据对象data都为undefined，还未初始化。在created阶段，vue实例的数据对象data有了，$el还没有。

**载入前/后：**在beforeMount阶段，vue实例的$el和data都初始化了，但还是挂载之前为虚拟的dom节点，data.message还未替换。在mounted阶段，vue实例挂载完成，data.message成功渲染。

**更新前/后：**当data变化时，会触发beforeUpdate和updated方法。

**销毁前/后：**在执行destroy方法后，对data的改变不会再触发周期函数，说明此时vue实例已经解除了事件监听以及和dom的绑定，但是dom结构依然存在

## 18、请说下封装 vue 组件的过程？

答：首先，组件可以提升整个项目的开发效率。能够把页面抽象成多个相对独立的模块，解决了我们传统项目开发：**效率低**、**难维护**、**复用性**等问题。

然后，使用Vue.extend方法创建一个组件，然后使用Vue.component方法注册组件。子组件需要数据，可以在props中接受定义。而子组件修改好数据后，想把数据传递给父组件。可以采用emit方法。

## 19、你是怎么认识vuex的？

答：vuex可以理解为一种开发模式或框架。比如PHP有thinkphp，java有spring等。  
通过状态（数据源）集中管理驱动组件的变化（好比spring的IOC容器对bean进行集中管理）。

应用级的状态集中放在store中； 改变状态的方式是提交mutations，这是个同步的事物； 异步逻辑应该封装在action中。

## 20、vue-loader是什么？使用它的用途有哪些？

答：解析.vue文件的一个加载器，跟template/js/style转换成js模块。

用途：js可以写es6、style样式可以scss或less、template可以加jade等

## 21、请说出vue.cli项目中src目录每个文件夹和文件的用法？

答：assets文件夹是放静态资源；components是放组件；router是定义路由相关的配置;view视图；app.vue是一个应用主组件；main.js是入口文件

## 22、vue.cli中怎样使用自定义的组件？有遇到过哪些问题吗？

答：第一步：在components目录新建你的组件文件（smithButton.vue），script一定要export default {

第二步：在需要用的页面（组件）中导入：import smithButton from ‘../components/smithButton.vue’

第三步：注入到vue的子组件的components属性上面,components:{smithButton}

第四步：在template视图view中使用，<smith-button>  </smith-button>  
问题有：smithButton命名，使用的时候则smith-button。

## 23、聊聊你对Vue.js的template编译的理解？

答：简而言之，就是先转化成AST树，再得到的render函数返回VNode（Vue的虚拟DOM节点）

详情步骤：

首先，通过compile编译器把template编译成AST语法树（abstract syntax tree 即 源代码的抽象语法结构的树状表现形式），compile是createCompiler的返回值，createCompiler是用以创建编译器的。另外compile还负责合并option。

然后，AST会经过generate（将AST语法树转化成render funtion字符串的过程）得到render函数，render的返回值是VNode，VNode是Vue的虚拟DOM节点，里面有（标签名、子节点、文本等等）

# 综合

## 1、规避javascript多人开发函数重名问题

命名空间

封闭空间

js模块化mvc（数据层、表现层、控制层）

seajs

变量转换成对象的属性

对象化

## 2、请说出三种减低页面加载时间的方法

压缩css、js文件

合并js、css文件，减少http请求

外部js、css文件放在最底下

减少dom操作，尽可能用变量替代不必要的dom操作

## 3、你所了解到的Web攻击技术

（1）XSS（Cross-Site Scripting，跨站脚本攻击）：指通过存在安全漏洞的Web网站注册用户的浏览器内运行非法的HTML标签或者JavaScript进行的一种攻击。  
（2）SQL注入攻击  
（3）CSRF（Cross-Site Request Forgeries，跨站点请求伪造）：指攻击者通过设置好的陷阱，强制对已完成的认证用户进行非预期的个人信息或设定信息等某些状态更新。

## 4、web前端开发，如何提高页面性能优化？

内容方面：

1.减少 HTTP 请求 (Make Fewer HTTP Requests)

2.减少 DOM 元素数量 (Reduce the Number of DOM Elements)

3.使得 Ajax 可缓存 (Make Ajax Cacheable)

针对CSS：

1.把 CSS 放到代码页上端 (Put Stylesheets at the Top)

2.从页面中剥离 JavaScript 与 CSS (Make JavaScript and CSS External)

3.精简 JavaScript 与 CSS (Minify JavaScript and CSS)

4.避免 CSS 表达式 (Avoid CSS Expressions)

针对JavaScript ：

1. 脚本放到 HTML 代码页底部 (Put Scripts at the Bottom)

2. 从页面中剥离 JavaScript 与 CSS (Make JavaScript and CSS External)

3. 精简 JavaScript 与 CSS (Minify JavaScript and CSS)

4. 移除重复脚本 (Remove Duplicate Scripts)

面向图片(Image)：

1.优化图片

2 不要在 HTML 中使用缩放图片

3 使用恰当的图片格式

4 使用 CSS Sprites 技巧对图片优化

## 5、前端开发中，如何优化图像？图像格式的区别？

优化图像：

1、不用图片，尽量用css3代替。比如说要实现修饰效果，如半透明、边框、圆角、阴影、渐变等，在当前主流浏览器中都可以用CSS达成。

2、 使用矢量图SVG替代位图。对于绝大多数图案、图标等，矢量图更小，且可缩放而无需生成多套图。现在主流浏览器都支持SVG了，所以可放心使用！

3.、使用恰当的图片格式。我们常见的图片格式有JPEG、GIF、PNG。

基本上，内容图片多为照片之类的，适用于JPEG。

而修饰图片通常更适合用无损压缩的PNG。

GIF基本上除了GIF动画外不要使用。且动画的话，也更建议用video元素和视频格式，或用SVG动画取代。

4、按照HTTP协议设置合理的缓存。

5、使用字体图标webfont、CSS Sprites等。

6、用CSS或JavaScript实现预加载。

7、WebP图片格式能给前端带来的优化。WebP支持无损、有损压缩，动态、静态图片，压缩比率优于GIF、JPEG、JPEG2000、PG等格式，非常适合用于网络等图片传输。

 图像格式的区别：

矢量图：图标字体，如 font-awesome；svg

位图：gif,jpg(jpeg),png

区别：

　　1、gif:是是一种无损，8位图片格式。具有支持动画，索引透明，压缩等特性。适用于做色彩简单(色调少)的图片，如logo,各种小图标icons等。

　　2、JPEG格式是一种大小与质量相平衡的压缩图片格式。适用于允许轻微失真的色彩丰富的照片，不适合做色彩简单(色调少)的图片，如logo,各种小图标icons等。

　　3、png:PNG可以细分为三种格式:PNG8，PNG24，PNG32。后面的数字代表这种PNG格式最多可以索引和存储的颜色值。

关于透明：PNG8支持索引透明和alpha透明;PNG24不支持透明;而PNG32在24位的PNG基础上增加了8位（256阶）的alpha通道透明;

优缺点：

　　1、能在保证最不失真的情况下尽可能压缩图像文件的大小。

　　2、对于需要高保真的较复杂的图像，PNG虽然能无损压缩，但图片文件较大，不适合应用在Web页面上。

## 6、浏览器是如何渲染页面的？

渲染的流程如下：

1.解析HTML文件，创建DOM树。

   自上而下，遇到任何样式（link、style）与脚本（script）都会阻塞（外部样式不阻塞后续外部脚本的加载）。

2.解析CSS。优先级：浏览器默认设置<用户设置<外部样式<内联样式<HTML中的style样式；

3.将CSS与DOM合并，构建渲染树（Render Tree）

4.布局和绘制，重绘（repaint）和重排（reflow）