前端面试题整理

面试

目录

$HTML， HTTP，web综合问题

- 1、前端需要注意哪些`SEO`
- 2、`<img>`的`title`和`alt`有什么区别
- 3、`HTTP`的几种请求方法用途
- 4、从浏览器地址栏输入`url`到显示页面的步骤
- 5、如何进行网站性能优化
- 6、HTTP状态码及其含义
- 7、语义化的理解
- 8、介绍一下你对浏览器内核的理解
- 9、`html5`有哪些新特性、移除了那些元素？
- 10、`HTML5`的离线储存怎么使用，工作原理能不能解释一下？
- 11、浏览器是怎么对`HTML5`的离线储存资源进行管理和加载的呢
- 12、请描述一下`cookies`，`sessionStorage`和`localStorage`的区别
- 13、`iframe`有那些缺点？
- 14、`WEB`标准以及W3C标准是什么?
- 15、`xhtml`和`html`有什么区别?
- 16、`Doctype`作用? 严格模式与混杂模式如何区分？它们有何意义?
- 17、行内元素有哪些？块级元素有哪些？ 空(`void`)元素有那些？行内元素和块级元素有什么区别？
- 18、`HTML`全局属性(`global attribute`)有哪些
- 19、`Canvas`和`SVG`有什么区别？
- 20、`HTML5` 为什么只需要写 `<!DOCTYPE HTML>?`
- 21、如何在页面上实现一个圆形的可点击区域？
- 22、网页验证码是干嘛的，是为了解决什么安全问题

$CSS部分

- 1、`css sprite`是什么,有什么优缺点
- 2、`display: none;`与`visibility: hidden`;的区别
- 3、`link`与`@import`的区别
- 4、什么是`FOUC?`如何避免
- 5、如何创建块级格式化上下文(`block formatting context`),`BFC`有什么用
- 7、清除浮动的几种方式，各自的优缺点
- 8、为什么要初始化`CSS`样式?
- 9、`css3`有哪些新特性
- 10、`display`有哪些值？说明他们的作用
- 11、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？
- 12、`CSS`优先级算法如何计算？
- 13、对`BFC`规范的理解？

- 14、谈谈浮动和清除浮动
- 15、`position`的值， `relative`和absolute`定位原点是
- 16、`display:inline-block` 什么时候不会显示间隙？(携程)
- 17、`PNG,GIF,JPG`的区别及如何选
- 18、行内元素`float:left`后是否变为块级元素？
- 19、在网页中的应该使用奇数还是偶数的字体？为什么呢？
- 20、`::before` 和 `:after`中双冒号和单冒号有什么区别？解释一下这`2`个伪元素的作用
- 21、如果需要手动写动画，你认为最小时间间隔是多久，为什么？（阿里）

$JavaScript

- 1、闭包
- 2、说说你对作用域链的理解
- 3、`JavaScript`原型，原型链？有什么特点？
- 4、请解释什么是事件代理
- 5、`Javascript`如何实现继承？
- 6、谈谈`This`对象的理解
- 7、事件模型
- 8、`new`操作符具体干了什么呢?
- 9、`Ajax`原理
- 11、模块化开发怎么做？
- 12、异步加载`JS`的方式有哪些？
- 13、那些操作会造成内存泄漏？
- 14、`XML`和`JSON`的区别？
- 15、谈谈你对`webpack`的看法
- 17、常见`web`安全及防护原理
- 18、用过哪些设计模式？
- 19、为什么要有同源限制？
- 20、`offsetWidth/offsetHeight`,`clientWidth/clientHeight`与`scrollWidth/scrollHeight`的区别
- 21、`javascript`有哪些方法定义对象
- 22、常见兼容性问题？
- 22、说说你对`promise`的了解
- 23、你觉得`jQuery`源码有哪些写的好的地方
- 25、`Node`的应用场景
- 26、谈谈你对`AMD`、`CMD`的理解
- 27、那些操作会造成内存泄漏？
- 28、`web`开发中会话跟踪的方法有哪些
- 29、介绍`js`的基本数据类型
- 30、介绍`js`有哪些内置对象？
- 31、说几条写`JavaScript`的基本规范？
- 32、`JavaScript`有几种类型的值？，你能画一下他们的内存图吗？
- 33、`javascript`创建对象的几种方式？
- 34、`eval`是做什么的？
- 35、`null, undefined` 的区别？
- 36、`["1", "2", "3"].map(parseInt)` 答案是多少？
- 37、`javascript` 代码中的`"use strict"`;是什么意思？使用它区别是什么？
- 38、`JSON`的了解？
- 39、js延迟加载的方式有哪些？

- 40、同步和异步的区别?
- 41、渐进增强和优雅降级
- 42、`defer`和`async`
- 43、说说严格模式的限制
- 44、`attribute`和`property`的区别是什么？
- 45、谈谈你对`ES6`的理解
- 46、ECMAScript6 怎么写class么，为什么会出现class这种东西?

$编程题

- 1、写一个通用的事件侦听器函数
- 2、如何判断一个对象是否为数组
- 3、冒泡排序
- 4、快速排序
- 5、编写一个方法 求一个字符串的字节长度

$其他

- 1、谈谈你对重构的理解
- 2、什么样的前端代码是好的
- 3、对前端工程师这个职位是怎么样理解的？它的前景会怎么样？
- 4、你觉得前端工程的价值体现在哪
- 5、平时如何管理你的项目？

人事面

- 面试完你还有什么问题要问的吗
- 你有什么爱好?
- 你最大的优点和缺点是什么?
- 你为什么会选择这个行业，职位?
- 你觉得你适合从事这个岗位吗?
- 你有什么职业规划?
- 你对工资有什么要求?
- 如何看待前端开发？
- 未来三到五年的规划是怎样的？

一些问题

- 你的项目中技术难点是什么？
- 遇到了什么问题？
- 你是怎么解决的？
- 最近在看哪些前端方面的书？
- 平时是如何学习前端开发的？
- 为什么大型应用需要从多个域名请求资源？
- 什么样的页面具有良好的用户体验？
- 是否了解`Web`注入攻击，说下原理，最- 常见的两种攻击（`XSS` 和 `CSRF`）了解到什么程度

$HTML， HTTP，web综合问题

1、前端需要注意哪些SEO

- 合理的`title`、`description`、`keywords`：搜索对着三项的权重逐个减小，title值强调重点即可，重要关键词出现不要超过2次，而且要靠前，不同页面`title`要有所不同；`description`把页面内容高度概括，长度合适，不可过分堆砌关键词，不同页面`description`有所不同；`keywords`列举出重要关键词即可
- 语义化的`HTML`代码，符合W3C规范：语义化代码让搜索引擎容易理解网页
- 重要内容`HTML`代码放在最前：搜索引擎抓取`HTML`顺序是从上到下，有的搜索引擎对抓取长度有限制，保证重要内容一定会被抓取
- 重要内容不要用`js`输出：爬虫不会执行js获取内容
- 少用`iframe`：搜索引擎不会抓取`iframe`中的内容
- 非装饰性图片必须加`alt`
- 提高网站速度：网站速度是搜索引擎排序的一个重要指标

2、`<img>`的`title`和`alt`有什么区别

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

3、HTTP的几种请求方法用途

- 1、`GET`方法
    - 发送一个请求来取得服务器上的某一资源
- 2、`POST`方法
    - 向`URL`指定的资源提交数据或附加新的数据
- 3、`PUT`方法
    - 跟`POST`方法很像，也是想服务器提交数据。但是，它们之间有不同。PUT指定了资源在服务器上的位置，而`POST`没有
- 4、`HEAD`方法
    - 只请求页面的首部
- 5、`DELETE`方法
    - 删除服务器上的某资源
- 6、`OPTIONS`方法
    - 它用于获取当前`URL`所支持的方法。如果请求成功，会有一个`Allow`的头包含类似`"GET,POST"`这样的信息
- 7、`TRACE`方法

    - TRACE方法被用于激发一个远程的，应用层的请求消息回路
- 8、CONNECT方法
    - 把请求连接转换到透明的TCP/IP通道

4、从浏览器地址栏输入url到显示页面的步骤

- 浏览器根据请求的URL交给DNS域名解析，找到真实IP，向服务器发起请求；
- 服务器交给后台处理完成后返回数据，浏览器接收文件（HTML、JS、CSS、图象等）；
- 浏览器对加载到的资源（HTML、JS、CSS等）进行语法解析，建立相应的内部数据结构（如HTML的DOM）；
- 载入解析到的资源文件，渲染页面，完成。

5、如何进行网站性能优化

- content方面
    1. 减少HTTP请求：合并文件、CSS精灵、inline Image
    2. 减少DNS查询：DNS缓存、将资源分布到恰当数量的主机名
    3. 减少DOM元素数量
- Server方面
    1. 使用CDN
    2. 配置ETag
    3. 对组件使用Gzip压缩
- Cookie方面
    1. 减小cookie大小
- css方面
    1. 将样式表放到页面顶部
    2. 不使用CSS表达式
    3. 使用<link>不使用@import
- Javascript方面
    1. 将脚本放到页面底部
    2. 将javascript和css从外部引入
    3. 压缩javascript和css
    4. 删除不需要的脚本
    5. 减少DOM访问
- 图片方面
    1. 优化图片：根据实际颜色需要选择色深、压缩
    2. 优化css精灵
    3. 不要在HTML中拉伸图片

6、HTTP状态码及其含义

- `1XX`：信息状态码
  - `100 Continue` 继续，一般在发送`post`请求时，已发送了`http header`之后服务端将返回此信息，表示确认，之后发送具体参数信息
- `2XX`：成功状态码
  - `200 OK` 正常返回信息
  - `201 Created` 请求成功并且服务器创建了新的资源
  - `202 Accepted` 服务器已接受请求，但尚未处理
- `3XX`：重定向
  - `301 Moved Permanently` 请求的网页已永久移动到新位置。
  - `302 Found` 临时性重定向。
  - `303 See Other` 临时性重定向，且总是使用 `GET` 请求新的 `URI`。
  - `304 Not Modified` 自从上次请求后，请求的网页未修改过。
- `4XX`：客户端错误
  - `400 Bad Request` 服务器无法理解请求的格式，客户端不应当尝试再次使用相同的内容发起请求。
  - `401 Unauthorized` 请求未授权。
  - `403 Forbidden` 禁止访问。
  - `404 Not Found` 找不到如何与 `URI` 相匹配的资源。
- `5XX:` 服务器错误
  - `500 Internal Server Error` 最常见的服务器端错误。
  - `503 Service Unavailable` 服务器端暂时无法处理请求（可能是过载或维护）。

7、语义化的理解

- 用正确的标签做正确的事情！
- `html`语义化就是让页面的内容结构化，便于对浏览器、搜索引擎解析；
- 在没有样式`CSS`情况下也以一种文档格式显示，并且是容易阅读的。
- 搜索引擎的爬虫依赖于标记来确定上下文和各个关键字的权重，利于 `SEO`。
- 使阅读源代码的人对网站更容易将网站分块，便于阅读维护理解

8、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

9、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`
- 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`

10、`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：

  - 页面头部像下面一样加入一个`manifest`的属性；
  - 在`cache.manifest`文件的编写离线存储的资源
  - 在离线状态时，操作`window.applicationCache`进行需求实现

```
CACHE MANIFEST
#v0.11
CACHE:
js/app.js
css/style.css
NETWORK:
resourse/logo.png
FALLBACK:
/ /offline.html
```

11、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。

12、请描述一下 cookies， sessionStorage 和 localStorage 的区别？

- cookie是网站为了标示用户身份而储存在用户本地终端（Client Side）上的数据（通常经过加密）

- cookie数据始终在同源的http请求中携带（即使不需要），记会在浏览器和服务器间来回传递

- sessionStorage和localStorage不会自动把数据发给服务器，仅在本地保存

- 存储大小：
  - cookie数据大小不能超过4k
  - sessionStorage和localStorage虽然也有存储大小的限制，但比cookie大得多，可以达到5M或更大
- 有期时间：
  - localStorage 存储持久数据，浏览器关闭后数据不丢失除非主动删除数据
  - sessionStorage 数据在当前浏览器窗口关闭后自动删除
  - cookie 设置的cookie过期时间之前一直有效，即使窗口或浏览器关闭

13、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

14、WEB标准以及W3C标准是什么？

- 标签闭合、标签小写、不乱嵌套、使用外链css和js、结构行为表现的分离

15、xhtml和html有什么区别?

- 一个是功能上的差别
    - 主要是`XHTML`可兼容各大浏览器、手机以及`PDA`，并且浏览器也能快速正确地编译网页
- 另外是书写习惯的差别
    - `XHTML` 元素必须被正确地嵌套，闭合，区分大小写，文档必须拥有根元素

16、Doctype作用? 严格模式与混杂模式如何区分？它们有何意义?

- 页面被加载的时，`link`会同时被加载，而`@imort`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`只在`IE5`以上才能识别，而`link`是`XHTML`标签，无兼容问题
  `link`方式的样式的权重 高于`@import`的权重
- `<!DOCTYPE>` 声明位于文档中的最前面，处于 `<html>` 标签之前。告知浏览器的解析器， 用什么文档类型 规范来解析这个文档
- 严格模式的排版和 `JS` 运作模式是 以该浏览器支持的最高标准运行
- 在混杂模式中，页面以宽松的向后兼容的方式显示。模拟老式浏览器的行为以防止站点无法工作。 `DOCTYPE`不存在或格式不正确会导致文档以混杂模式呈现

17、行内元素有哪些？块级元素有哪些？ 空(void)元素有那些？行内元素和块级元素有什么区别？

- 行内元素有：`a b span img input select strong`
- 块级元素有：`div ul ol li dl dt dd h1 h2 h3 h4…p`
- 空元素：`<br> <hr> <img> <input> <link> <meta>`
- 行内元素不可以设置宽高，不独占一行
- 块级元素可以设置宽高，独占一行

18、HTML全局属性(global attribute)有哪些

- `class`:为元素设置类标识
- `data-*`: 为元素增加自定义属性
- `draggable`: 设置元素是否可拖拽
- `id`: 元素`id`，文档内唯一
- `lang`: 元素内容的的语言
- `style`: 行内`css`样式
- `title`: 元素相关的建议信息

19、Canvas和SVG有什么区别？

- `svg`绘制出来的每一个图形的元素都是独立的`DOM`节点，能够方便的绑定事件或用来修改。`canvas`输出的是一整幅画布
- `svg`输出的图形是矢量图形，后期可以修改参数来自由放大缩小，不会是真和锯齿。而`canvas`输出标量画布，就像一张图片一样，放大会失真或者锯齿

20、HTML5 为什么只需要写 ？

- `HTML5` 不基于 `SGML`，因此不需要对`DTD`进行引用，但是需要`doctype`来规范浏览器的行为
- 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

21、如何在页面上实现一个圆形的可点击区域？

- `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

22、网页验证码是干嘛的，是为了解决什么安全问题

- 区分用户是计算机还是人的公共全自动程序。可以防止恶意破解密码、刷票、论坛灌水
- 有效防止黑客对某一个特定注册用户用特定程序暴力破解方式进行不断的登陆尝试

$CSS部分

1、css sprite是什么,有什么优缺点

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

- 优点：

  - 减少`HTTP`请求数，极大地提高页面加载速度
  - 增加图片信息重复度，提高压缩比，减少图片大小
  - 更换风格方便，只需在一张或几张图片上修改颜色或样式即可实现
- 缺点：

  - 图片合并麻烦
  - 维护麻烦，修改一个图片可能需要从新布局整个图片，样式

2、`display: none;`与`visibility: hidden;`的区别

- 联系：它们都能让元素不可见

- 区别：

  - `display:none`;会让元素完全从渲染树中消失，渲染的时候不占据任何空间；`visibility: hidden`;不会让元素从渲染树消失，渲染师元素继续占据空间，只是内容不可见
  - `display: none`;是非继承属性，子孙节点消失由于元素从渲染树消失造成，通过

修改子孙节点属性无法显示`;visibility: hidden;`是继承属性，子孙节点消失由于继承了`hidden`，通过设置`visibility: visible;`可以让子孙节点显式
    - 修改常规流中元素的`display`通常会造成文档重排。修改`visibility`属性只会造成本元素的重绘。
    - 读屏器不会读取`display: none`;元素内容；会读取`visibility: hidden;`元素内容

3、`link`与`@import`的区别

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

4、什么是FOUC?如何避免

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

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

- 创建规则：
    - 根元素
    - 浮动元素（`float`不是`none`）
    - 绝对定位元素（`position`取值为`absolute`或`fixed`）
    - `display`取值为`inline-block`,`table-cell`,`table-caption`,`flex`,`inline-flex`之一的元素
    - `overflow`不是`visible`的元素
- 作用：
    - 可以包含浮动元素
    - 不被浮动元素覆盖
    - 阻止父子元素的`margin`折叠

6、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`**

7、清除浮动的几种方式，各自的优缺点

- 使用空标签清除浮动`clear:both`（缺点，增加无意义的标签）
- 使用`overflow:auto`（使用`zoom:1`用于兼容`IE`，缺点：内部宽高超过父级`div`时，会出现滚动条）
- 用`afert`伪元素清除浮动(`IE8`以上和非`IE`浏览器才支持，目前：大型网站都有使用

8、为什么要初始化CSS样式?

- 因为浏览器的兼容问题，不同浏览器对有些标签的默认值是不同的，如果没对`CSS`初始化往往会出现浏览器之间的页面显示差异。
- 当然，初始化样式会对`SEO`有一定的影响，但鱼和熊掌不可兼得，但力求影响最小的情况下初始化

9、css3有哪些新特性

- 新增各种`css`选择器
- 圆角 `border-radius`
- 多列布局
- 阴影和反射
- 文字特效`text-shadow`
- 线性渐变
- 旋转`transform`

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`
- `:disabled` 控制表单控件的禁用状态。
- `:checked` 单选框或复选框被选中

10、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

11、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`);
- 区 别： `IE`的c`ontent`部分把 `border` 和 `padding`计算了进去;

12、CSS优先级算法如何计算？

- 优先级就近原则，同权重情况下样式定义最近者为准
- 载入样式以最后载入的定位为准
- 优先级为: `!important > id > class > tag` `important` 比 内联优先级高

13、对BFC规范的理解？

- 它决定了元素如何对其内容进行定位,以及与其他元素的关系和相互作用

14、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

15、position的值， relative和absolute定位原点是

- `absolute`：生成绝对定位的元素，相对于 `static` 定位以外的第一个父元素进行定位
- `fixed`： 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
- `inherit` 规定从父元素继承 `position` 属性的值

16、display:inline-block 什么时候不会显示间隙？(携程)

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

17、PNG,GIF,JPG的区别及如何选

- `GIF`
    - `8`位像素， `256`色
    - 无损压缩
    - 支持简单动画
    - 支持`boolean`透明
    - 适合简单动画
- `JPEG`
    - 颜色限于`256`
    - 有损压缩

    - 可控制压缩质量
    - 不支持透明
    - 适合照片
- `PNG`
    - 有`PNG8`和`truecolor PNG`
    - `PNG8`类似`GIF`颜色上限为`256`，文件小，支持`alpha`透明度，无动画
    - 适合图标、背景、按钮

18、行内元素float:left后是否变为块级元素？

- 浮动后，行内元素不会成为块状元素，但是可以设置宽高。行内元素要想变成块状元素，占一行，直接设置`display:block`;。但如果元素设置了浮动后再设置`display:block`;那就不会占一行。

19、在网页中的应该使用奇数还是偶数的字体？为什么呢？

- 偶数字号相对更容易和 web 设计的其他部分构成比例关系

20、::before 和 :after中双冒号和单冒号 有什么区别？解释一下这2个伪元素的作用

- 单冒号(`:`)用于CSS3伪类，双冒号(`::`)用于`CSS3`伪元素
- 用于区分伪类和伪元素

21、如果需要手动写动画，你认为最小时间间隔是多久，为什么？（阿里）

- 多数显示器默认频率是`60Hz`，即`1`秒刷新`60`次，所以理论上最小间隔为`1/60*1000ms = 16.7ms`

$JavaScript

1、闭包

- 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

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 的最大用处有两个，一个是可以读取函数内部的变量，另一个就是让这些变量始终保持在内存中，即闭包可以使得它诞生环境一直存在

- 闭包的另一个用处，是封装对象的私有属性和私有方法

2、说说你对作用域链的理解

- 作用域链的作用是保证执行环境里有权访问的变量和函数是有序的，作用域链的变量只能向上访问，变量访问到`window`对象即被终止，作用域链向下访问变量是不被允许的
- 简单的说，作用域就是变量与函数的可访问范围，即作用域控制着变量与函数的可见性和生命周期

3、JavaScript原型，原型链？有什么特点？

- 每个对象都会在其内部初始化一个属性，就是`prototype`(原型)，当我们访问一个对象的属性时

- 如果这个对象内部不存在这个属性，那么他就会去`prototype`里找这个属性，这`个prototype`又会有自己的`prototype`，于是就这样一直找下去，也就是我们平时所说的原型链的概念

- 关系：`instance.constructor.prototype = instance.__proto__`

- 特点：

    - `JavaScript`对象是通过引用来传递的，我们创建的每个新对象实体中并没有一份属于自己的原型副本。当我们修改原型时，与之相关的对象也会继承这一改变
- 当我们需要一个属性的时，`Javascript`引擎会先看当前对象中是否有这个属性， 如果没有的

- 就会查找他的`Prototype`对象是否有这个属性，如此递推下去，一直检索到 `Object` 内建对象

4、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

- 可以大量节省内存占用，减少事件注册，比如在`table`上代理所有`td`的`click`事件就非常棒
- 可以实现当新增子对象时无需再次对其绑定

5、Javascript如何实现继承？

- 构造继承
- 原型继承
- 实例继承
- 拷贝继承
- 原型`prototype`机制或`apply`和`call`方法去实现较简单，建议使用构造函数与原型混合方式

```
function Parent(){
        this.name = 'wang';
    }

    function Child(){
        this.age = 28;
    }
    Child.prototype = new Parent();//继承了Parent, 通过原型

    var demo = new Child();
    alert(demo.age);
    alert(demo.name);//得到被继承的属性
 }
```

6、谈谈This对象的理解

- `this`总是指向函数的直接调用者（而非间接调用者）
- 如果有`new`关键字，`this`指向`new`出来的那个对象
- 在事件中，`this`指向触发这个事件的对象，特殊的是，`IE`中的`attachEvent`中的`this`总是指向全局对象`Window`

7、事件模型

- 冒泡型事件：当你使用事件冒泡时，子级元素先触发，父级元素后触发
- 捕获型事件：当你使用事件捕获时，父级元素先触发，子级元素后触发
- `DOM`事件流：同时支持两种事件模型：捕获型事件和冒泡型事件
- 阻止冒泡：在`W3c`中，使用`stopPropagation()`方法；在IE下设置`cancelBubble = true`
- 阻止捕获：阻止事件的默认行为，例如`click - <a>`后的跳转。在`W3c`中，使用`preventDefault()`方法，在`IE`下设置`window.event.returnValue = false`

8、new操作符具体干了什么呢?

- 创建一个空对象，并且 `this` 变量引用该对象，同时还继承了该函数的原型
- 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`

9、Ajax原理

- `Ajax` 的原理简单来说是在用户和服务器之间加了—个中间层(`AJAX` 引擎)，通过 `XmlHttpRequest` 对象来向服务器发异步请求，从服务器获得数据，然后用 `javascrip` t来操作 `DOM` 而更新页面。使用户操作与服务器响应异步化。这其中最关键的一步就是从服务器获得请求数据
- `Ajax` 的过程只涉及 `JavaScript` 、 `XMLHttpRequest` 和 `DOM` 。 `XMLHttpRequest` 是 `aja` x的核心机制

```
// 1. 创建连接
    var xhr = null;
    xhr = new XMLHttpRequest()
    // 2. 连接服务器
    xhr.open('get', url, true)
    // 3. 发送请求
    xhr.send(null);
    // 4. 接受请求
    xhr.onreadystatechange = function(){
        if(xhr.readyState == 4){
            if(xhr.status == 200){
                success(xhr.responseText);
            } else { // fail
                fail && fail(xhr.status);
            }
        }
    }
```

10、如何解决跨域问题?

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

11、模块化开发怎么做?

- 立即执行函数,不暴露私有成员

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

12、异步加载JS的方式有哪些？

- defer，只支持`IE`
- `async`：
- 创建`script`，插入到`DOM`中，加载完毕后`callBack`

13、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
- `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
- 闭包使用不当

14、XML和JSON的区别？

- 数据体积方面
  - `JSON`相对`于XML`来讲，数据的体积小，传递的速度更快些。
- 数据交互方面
  - `JSON`与`JavaScript`的交互更加方便，更容易解析处理，更好的数据交互
- 数据描述方面
  - `JSON`对数据的描述性比`XML`较差
- 传输速度方面
  - `JSON`的速度要远远快于`XML`

15、谈谈你对webpack的看法

- `WebPack` 是一个模块打包工具，你可以使用`WebPack`管理你的模块依赖，并编绎输出模块们所需的静态文件。它能够很好地管理、打包`Web`开发中所用到的`HTML`、`Javascript`、`CSS`以及各种静态文件（图片、字体等），让开发过程更加高效。对于不同类型的资源，`webpack`有对应的模块加载器。`webpack`模块打包器会分析模块间的依赖关系，最后 生成了优化且合并后的静态资源

16、说说你对AMD和Commonjs的理解

- `CommonJS`是服务器端模块的规范，`Node.js`采用了这个规范。`CommonJS`规范加载模块是同步的，也就是说，只有加载完成，才能执行后面的操作。`AMD`规范则是非同步加载模块，允许指定回调函数
- `AMD`推荐的风格通过返回一个对象做为模块对象，`CommonJS`的风格通过对`module.exports`或`exports`的属性赋值来达到暴露模块对象的目的

17、常见web安全及防护原理

- `sql`注入原理
  - 就是通过把`SQL`命令插入到`Web`表单递交或输入域名或页面请求的查询字符串，

最终达到欺骗服务器执行恶意的SQL命令

- 总的来说有以下几点
  - 永远不要信任用户的输入，要对用户的输入进行校验，可以通过正则表达式，或限制长度，对单引号和双`"-"`进行转换等
  - 永远不要使用动态拼装SQL，可以使用参数化的`SQL`或者直接使用存储过程进行数据查询存取
  - 永远不要使用管理员权限的数据库连接，为每个应用使用单独的权限有限的数据库连接
  - 不要把机密信息明文存放，请加密或者`hash`掉密码和敏感的信息

XSS原理及防范

- `Xss(cross-site scripting)`攻击指的是攻击者往`Web`页面里插入恶意`html`标签或者`javascript`代码。比如：攻击者在论坛中放一个看似安全的链接，骗取用户点击后，窃取`cookie`中的用户私密信息；或者攻击者在论坛中加一个恶意表单，当用户提交表单的时候，却把信息传送到攻击者的服务器中，而不是用户原本以为的信任站点

XSS防范方法

- 首先代码里对用户输入的地方和变量都需要仔细检查长度和对`"<",">",";","'"`等字符做过滤；其次任何内容写到页面之前都必须加以encode，避免不小心把`html tag`弄出来。这一个层面做好，至少可以堵住超过一半的XSS 攻击

XSS与CSRF有什么区别吗？

- `XSS`是获取信息，不需要提前知道其他用户页面的代码和数据包。`CSRF`是代替用户完成指定的动作，需要知道其他用户页面的代码和数据包。要完成一次`CSRF`攻击，受害者必须依次完成两个步骤

- 登录受信任网站`A`，并在本地生成`Cookie`

- 在不登出`A`的情况下，访问危险网站`B`

CSRF的防御

- 服务端的`CSRF`方式方法很多样，但总的思想都是一致的，就是在客户端页面增加伪随机数
- 通过验证码的方法

18、用过哪些设计模式？

- 工厂模式：
  - 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
  - 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
- 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
- 直接将属性和方法赋值给 `this` 对象;

19、为什么要有同源限制？

- 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
- 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。

20、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**

21、javascript有哪些方法定义对象

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

22、常见兼容性问题？

- `png24` 位的图片在iE6浏览器上出现背景，解决方案是做成 `PNG8`
- 浏览器默认的 `margin` 和 `padding` 不同。解决方案是加一个全局的 `*{margin:0;padding:0;}` 来统一,，但是全局效率很低，一般是如下这样解决：

```
body,ul,li,ol,dl,dt,dd,form,input,h1,h2,h3,h4,h5,h6,p{
margin:0;
padding:0;
}
```

- `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.

22、说说你对promise的了解

- 依照 `Promise/A+` 的定义， `Promise` 有四种状态：
  - `pending:` 初始状态, 非 `fulfilled` 或 `rejected.`
  - `fulfilled:` 成功的操作.

- `rejected:` 失败的操作.
- `settled: Promise` 已被 `fulfilled` 或 `rejected`，且不是 `pending`

- 另外， `fulfilled` 与 `rejected` 一起合称 `settled`
- `Promise` 对象用来进行延迟(`deferred`) 和异步(`asynchronous`) 计算

Promise 的构造函数

- 构造一个 `Promise`，最基本的用法如下：

```
var promise = new Promise(function(resolve, reject) {
        if (...) {  // succeed
            resolve(result);
        } else {   // fails
            reject(Error(errMessage));
        }
    });
```

- `Promise` 实例拥有 `then` 方法（具有 `then` 方法的对象，通常被称为 `thenable` ）。它的使用方法如下：

```
promise.then(onFulfilled, onRejected)
```

- 接收两个函数作为参数，一个在 `fulfilled` 的时候被调用，一个在 `rejected` 的时候被调用，接收参数就是 `future`， `onFulfilled` 对应 `resolve`，`onRejected` 对应 `reject`

23、你觉得jQuery源码有哪些写的好的地方

- `jquery` 源码封装在一个匿名函数的自执行环境中，有助于防止变量的全局污染，然后通过传入 `window` 对象参数，可以使 `window` 对象作为局部变量使用，好处是当 `jquery` 中访问 `window` 对象的时候，就不用将作用域链退回到顶层作用域了，从而可以更快的访问window对象。同样，传入 `undefined` 参数，可以缩短查找 `undefined` 时的作用域链
- `jquery` 将一些原型属性和方法封装在了 `jquery.prototype` 中，为了缩短名称，又赋值给了 `jquery.fn`，这是很形象的写法
- 有一些数组或对象的方法经常能使用到， `jQuery` 将其保存为局部变量以提高访问速度
- `jquery` 实现的链式调用可以节约代码，所返回的都是同一个对象，可以提高代码效率

24、vue、react、angular

- `Vue.js`

一个用于创建 web 交互界面的库，是一个精简的 MVVM。它通过双向数据绑定把 View 层和 Model 层连接了起来。实际的 DOM 封装和输出格式都被抽象为了Directives 和 Filters

- AngularJS
  是一个比较完善的前端MVVM框架，包含模板，数据双向绑定，路由，模块化，服务，依赖注入等所有功能，模板功能强大丰富，自带了丰富的 Angular指令

- react
  React 仅仅是 VIEW 层是facebook公司。推出的一个用于构建UI的一个库，能够实现服务器端的渲染。用了virtual dom，所以性能很好。

25、Node的应用场景

- 特点：
  - 1、它是一个Javascript运行环境
  - 2、依赖于Chrome V8引擎进行代码解释
  - 3、事件驱动
  - 4、非阻塞I/O
  - 5、单进程，单线程
- 优点：
  - 高并发（最重要的优点）
- 缺点：
  - 1、只支持单核CPU，不能充分利用CPU
  - 2、可靠性低，一旦代码某个环节崩溃，整个系统都崩溃

26、谈谈你对AMD、CMD的理解

- CommonJS是服务器端模块的规范，Node.js采用了这个规范。CommonJS规范加载模块是同步的，也就是说，只有加载完成，才能执行后面的操作。AMD规范则是非同步加载模块，允许指定回调函数

- AMD推荐的风格通过返回一个对象做为模块对象，CommonJS的风格通过对module.exports或exports的属性赋值来达到暴露模块对象的目的

27、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
- 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）

28、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

29、介绍js的基本数据类型

- `Undefined`、`Null`、`Boolean`、`Number`、`String`

30、介绍js有哪些内置对象？

- `Object` 是 `JavaScript` 中所有对象的父对象
- 数据封装类对象：`Object`、`Array`、`Boolean`、`Number` 和 `String`
- 其他对象：`Function`、`Arguments`、`Math`、`Date`、`RegExp`、`Error`

31、说几条写JavaScript的基本规范？

- 不要在同一行声明多个变量
- 请使用 `===/!==` 来比较 `true/false` 或者数值
- 使用对象字面量替代 `new Array` 这种形式
- 不要使用全局函数
- `Switch` 语句必须带有 `default` 分支
- `If` 语句必须使用大括号
- `for-in` 循环中的变量 应该使用 `var` 关键字明确限定作用域，从而避免作用域污

32、JavaScript有几种类型的值？，你能画一下他们的内存图吗？

- 栈：原始数据类型（`Undefined`，`Null`，`Boolean`，`Numbe`r、`String`）
- 堆：引用数据类型（对象、数组和函数）
- 两种类型的区别是：存储位置不同；
- 原始数据类型直接存储在栈(`stack`)中的简单数据段，占据空间小、大小固定，属于被频繁使用数据，所以放入栈中存储；
- 引用数据类型存储在堆(`heap`)中的对象,占据空间大、大小不固定,如果存储在栈中，将会影响程序运行的性能；引用数据类型在栈中存储了指针，该指针指向堆中该实体的起始地址。当解释器寻找引用值时，会首先检索其
- 在栈中的地址，取得地址后从堆中获得实体

33、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

- 对象字面量的方式

```
person={firstname:"Mark",lastname:"Yun",age:25,eyecolor:"black"};
```

- 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Cl
ass
        person.name="Mark";
        person.age="25";
        person.work=function(){
        alert(person.name+" hello...");
    }
person.work();
```

- 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
```

- 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();
```

- 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();
```

- 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();
```

34、eval是做什么的？

- 它的功能是把对应的字符串解析成`JS`代码并运行
- 应该避免使用`eval`，不安全，非常耗性能（`2`次，一次解析成`js`语句，一次执行）
- 由`JSON`字符串转换为JSON对象的时候可以用`eval, var obj =eval('('+ str +')')`

35、null，undefined 的区别？

- `undefined` 表示不存在这个值。
- `undefined` :是一个表示"无"的原始值或者说表示"缺少值"，就是此处应该有一个值，但是还没有定义。当尝试读取时会返回 `undefined`
- 例如变量被声明了，但没有赋值时，就等于`undefined`
- `null` 表示一个对象被定义了，值为“空值”
- `null` ：是一个对象(空对象, 没有任何属性和方法)
- 例如作为函数的参数，表示该函数的参数不是对象；
- 在验证`null`时，一定要使用 `===`，因为 `==`无法分别`null` 和 `undefined`

36、["1", "2", "3"].map(parseInt) 答案是多少？

- `[1, NaN, NaN]`因为 `parseInt` 需要两个参数 `(val, radix)`，其中`radix` 表示解析时用的基数。
- `map`传了 `3`个`(element, index, array)`，对应的 `radix` 不合法导致解析失败。

37、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

38、JSON 的了解？**

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

```
var obj =eval('('+ str +')');
var obj = str.parseJSON();
var obj = JSON.parse(str);
```

- `JSON`对象转换为JSON字符串：

```
var last=obj.toJSONString();
var last=JSON.stringify(obj);
```

39、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`

40、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

41、渐进增强和优雅降级

- 渐进增强 ：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 ：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

42、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

43、说说严格模式的限制

- 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

- 禁止`this`指向全局对象

44、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

45、谈谈你对ES6的理解

- 新增模板字符串（为`JavaScript`提供了简单的字符串插值功能）
- 箭头函数
- `for-of`（用来遍历数据—例如数组中的值。）
- `arguments`对象可被不定参数和默认参数完美代替。
- `ES6`将p`romise`对象纳入规范，提供了原生的`Promise`对象。
- 增加了`let`和`const`命令，用来声明变量。
- 增加了块级作用域。
- `let`命令实际上就增加了块级作用域。
- 还有就是引入`module`模块的概念

46、ECMAScript6 怎么写class么，为什么会出现class这种东西?

- 这个语法糖可以让有`OOP`基础的人更快上手`js`，至少是一个官方的实现了
- 但对熟悉`js`的人来说，这个东西没啥大影响；一个`Object.creat()`搞定继承，比`class`简洁清晰的多

$编程题

1、写一个通用的事件侦听器函数

```
// event(事件)工具集，来源：github.com/markyun
    markyun.Event = {

        // 视能力分别使用dom0||dom2||IE方式 来绑定事件
        // 参数： 操作的元素,事件名称 ,事件处理程序
        addEvent : function(element, type, handler) {
            if (element.addEventListener) {
                //事件类型、需要执行的函数、是否捕捉
                element.addEventListener(type, handler, false);
            } else if (element.attachEvent) {
                element.attachEvent('on' + type, function() {
                    handler.call(element);
                });
            } else {
                element['on' + type] = handler;
            }
        },
        // 移除事件
        removeEvent : function(element, type, handler) {
            if (element.removeEventListener) {
                element.removeEventListener(type, handler, false);
            } else if (element.datachEvent) {
                element.detachEvent('on' + type, handler);
            } else {
                element['on' + type] = null;
            }
        },
        // 阻止事件 (主要是事件冒泡，因为IE不支持事件捕获)
        stopPropagation : function(ev) {
            if (ev.stopPropagation) {
                ev.stopPropagation();
            } else {
                ev.cancelBubble = true;
            }
        },
        // 取消事件的默认行为
        preventDefault : function(event) {
            if (event.preventDefault) {
                event.preventDefault();
            } else {
                event.returnValue = false;
            }
        },
        // 获取事件目标
        getTarget : function(event) {
            return event.target || event.srcElement;
        }
```

2、如何判断一个对象是否为数组

```
function isArray(arg) {
    if (typeof arg === 'object') {
        return Object.prototype.toString.call(arg) === '[object Array]';
    }
    return false;
}
```

3、冒泡排序

- 每次比较相邻的两个数，如果后一个比前一个小，换位置

```
var arr = [3, 1, 4, 6, 5, 7, 2];

function bubbleSort(arr) {
for (var i = 0; i < arr.length - 1; i++) {
    for(var j = 0; j < arr.length - 1; j++) {
        if(arr[j + 1] < arr[j]) {
            var temp;
            temp = arr[j];
            arr[j] = arr[j + 1];
            arr[j + 1] = temp;
        }
    }
}
return arr;
}

console.log(bubbleSort(arr));
```

4、快速排序

- 采用二分法，取出中间数，数组每次和中间数比较，小的放到左边，大的放到右边

```
var arr = [3, 1, 4, 6, 5, 7, 2];

function quickSort(arr) {
    if(arr.length == 0) {
        return [];    // 返回空数组
    }

    var cIndex = Math.floor(arr.length / 2);
    var c = arr.splice(cIndex, 1);
    var l = [];
    var r = [];

    for (var i = 0; i < arr.length; i++) {
        if(arr[i] < c) {
            l.push(arr[i]);
        } else {
            r.push(arr[i]);
        }
    }

    return quickSort(l).concat(c, quickSort(r));
}

console.log(quickSort(arr));
```

5、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"));
```

$其他

1、谈谈你对重构的理解

- 网站重构：在不改变外部行为的前提下，简化结构、添加可读性，而在网站前端保持一致的行为。也就是说是在不改变UI的情况下，对网站进行优化， 在扩展的同时保持一致的UI
- 对于传统的网站来说重构通常是：
    - 表格(`table`)布局改为`DIV+CSS`
    - 使网站前端兼容于现代浏览器(针对于不合规范的`CSS`、如对IE6有效的)
    - 对于移动平台的优化
    - 针对于`SEO`进行优化

2、什么样的前端代码是好的

- 高复用低耦合，这样文件小，好维护，而且好扩展。

3、对前端工程师这个职位是怎么样理解的？它的前景会怎么样？

- 前端是最贴近用户的程序员，比后端、数据库、产品经理、运营、安全都近
    - 实现界面交互
    - 提升用户体验

    - 有了Node.js，前端可以实现服务端的一些事情
- 前端是最贴近用户的程序员，前端的能力就是能让产品从 90分进化到 100 分，甚至更好，
- 与团队成员，`UI`设计，产品经理的沟通；
- 做好的页面结构，页面重构和用户体验；

4、你觉得前端工程的价值体现在哪

- 为简化用户使用提供技术支持（交互部分）
- 为多个浏览器兼容性提供支持
- 为提高用户浏览速度（浏览器性能）提供支持
- 为跨平台或者其他基于webkit或其他渲染引擎的应用提供支持
- 为展示数据提供支持（数据接口）

5、平时如何管理你的项目？

- 先期团队必须确定好全局样式（`globe.css`），编码模式(`utf-8`) 等；
- 编写习惯必须一致（例如都是采用继承式的写法，单样式都写成一行）；
- 标注样式编写人，各模块都及时标注（标注关键样式调用的地方）；
- 页面进行标注（例如 页面 模块 开始和结束）；
- `CSS`跟`HTML` 分文件夹并行存放，命名都得统一（例如`style.css`）；
- `JS` 分文件夹存放 命名以该`JS`功能为准的英文翻译。
- 图片采用整合的 `images.png png8` 格式文件使用 - 尽量整合在一起使用方便将来的管理

人事面

- 面试完你还有什么问题要问的吗
- 你有什么爱好?
- 你最大的优点和缺点是什么?
- 你为什么会选择这个行业，职位?

- 你觉得你适合从事这个岗位吗?
- 你有什么职业规划?
- 你对工资有什么要求?
- 如何看待前端开发?
- 未来三到五年的规划是怎样的?

一些问题

- 你的项目中技术难点是什么?
- 遇到了什么问题?
- 你是怎么解决的?
- 最近在看哪些前端方面的书?
- 平时是如何学习前端开发的?
- 为什么大型应用需要从多个域名请求资源?
- 什么样的页面具有良好的用户体验?
- 是否了解`Web`注入攻击，说下原理，最常见的两种攻击`(XSS`和`CSRF`）了解到什么程度