面试宝典

一、HTML+CSS

1. Doctype 作用？标准模式与兼容模式各有什么区别?

（1）<!DOCTYPE>声明位于位于 HTML 文档中的第一行，处于 <html> 标签之前。告知浏览器的解析器用什么文档标准解析这个文档。

DOCTYPE 不存在或格式不正确会导致文档以兼容模式呈现。

（2）标准模式的排版 和 JS 运作模式都是以该浏览器支持的最高标准运行。在兼容模式中,页面以宽松的向后兼容的方式显示,模拟老式浏览器的行为以防止站点无法工作。

2. 行内元素有哪些？块级元素有哪些？ 空(void)元素有那些？

首先：CSS 规范规定，每个元素都有 display 属性，确定该元素的类型，每个元素都有默认的 display 值，如 div 的 display 默认值为“block”，则为“块级”元素；

span 默认 display 属性值为“inline”，是“行内”元素。

（1）行内元素有：a b span img input select strong（强调的语气）

（2）块级元素有：div ul ol li dl dt dd h1 h2 h3 h4…p

（3）常见的空元素：

 <hr> <img> <input> <link> <meta>

鲜为人知的是：

<area> <base> <col> <command> <embed> <keygen> <param> <source> <track> <wbr>

3. 页面导入样式时，使用 link 和@import 有什么区别？（腾讯）

（1）link 属于 XHTML 标签，除了加载 CSS 外，还能用于定义 RSS, 定义 rel 连接属性等作用；而@import 是 CSS 提供的，只能用于加载 CSS;

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

（3）import 是 CSS2.1 提出的，只在 IE5 以上才能被识别，而 link 是 XHTML 标签，无兼容问题;

4. 介绍一下你对浏览器内核的理解？（cvte）

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。

5. 简述一下你对 HTML 语义化的理解？（cvte）

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。

6.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>
```

7.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>
```

8.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。

9.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：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。

10. 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？

（1）有两种， IE 盒子模型、W3C 盒子模型；

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

（3）区 别： IE 的 content 部分把 border 和 padding 计算了进去;

11. CSS 选择符有哪些？哪些属性可以继承？

1.id 选择器（ # myid）

2.类选择器（.myclassname）

3.标签选择器（div, h1, p）

4.相邻选择器（h1 + p）

5.子选择器（ul > li）

6.后代选择器（li a）

7.通配符选择器（ * ）

8.属性选择器（a[rel = "external"]）

9.伪类选择器（a:hover, li:nth-child）

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

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

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:

同权重: 内联样式表（标签内部）> 嵌入样式表（当前文件中）> 外部样式表（外部文件中）。

!important > id > class > tag

important 比 内联优先级高

13. 如何居中 div？

水平居中：给 div 设置一个宽度，然后添加 margin:0 auto 属性

```
div{width:200px;margin:0 auto;}
```

让绝对定位的 div 居中

```
div {position: absolute;width: 300px;height: 300px;margin: auto;top: 0;
    left: 0;bottom: 0;right: 0;background-color: pink; /* 方便看效果 */}
```

水平垂直居中一

确定容器的宽高 宽 500 高 300 的层,设置层的外边距

```
div {position: relative;        /* 相对定位或绝对定位均可 */
    width:500px;height:300px;top: 50%;left: 50%;margin: -150px 0 0 -250px;
        /* 外边距为自身宽高的一半 */
    background-color: pink;        /* 方便看效果 */}
```

水平垂直居中二

未知容器的宽高，利用 `transform` 属性

```
div {position: absolute;        /* 相对定位或绝对定位均可 */
width:500px;height:300px;top: 50%;left: 50%;transform: translate(-50%, -50%);
background-color: pink;        /* 方便看效果 */}
```

水平垂直居中三

利用 flex 布局,实际使用时应考虑兼容性

```
.container {display: flex; align-items: center;/* 垂直居中 */justify-content: center;/* 水平居中 */}

.container div {width: 100px;height: 100px;background-color: pink;/* 方便看效果 */}
```

14. display 有哪些值？说明他们的作用。

block	块类型。默认宽度为父元素宽度，可设置宽高，换行显示。
none	缺省值。象行内元素类型一样显示。
inline	行内元素类型。默认宽度为内容宽度，不可设置宽高，同行显示。

inline-block　默认宽度为内容宽度，可以设置宽高，同行显示。

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。

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。

16. 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

17. css 多列等高如何实现？

利用 padding-bottom|margin-bottom 正负值相抵；

设置父容器设置超出隐藏（overflow:hidden），这样子父容器的高度就还是它里面的列没有设定 padding-bottom 时的高度，

当它里面的任 一列高度增加了，则父容器的高度被撑到里面最高那列的高度，

其他比这列矮的列会用它们的 padding-bottom 补偿这部分高度差。

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

png24 位的图片在 iE6 浏览器上出现背景，解决方案是做成 PNG8.

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

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

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

这种情况之下 IE 会产生 20px 的距离，解决方案是在 float 的标签样式控制中加入 ——_display:inline;将其转化为行内属性。(_这个符号只有 ie6 会识别)

渐进识别的方式，从总体中逐渐排除局部。

首先，巧妙的使用“\9”这一标记，将 IE 游览器从所有情况中分离出来。

接着，再次使用“+”将 IE8 和 IE7、IE6 分离开来，这样 IE8 已经独立识别。

```
css
    .bb{
        background-color:red;/*所有识别*/
        background-color:#00deff\9; /*IE6、7、8 识别*/
        +background-color:#a200ff;/*IE6、7 识别*/
        _background-color:#1e0bd1;/*IE6 识别*/
    }
```

IE 下,可以使用获取常规属性的方法来获取自定义属性,

也可以使用 getAttribute()获取自定义属性;

Firefox 下,只能使用 getAttribute()获取自定义属性。

解决方法:统一通过 getAttribute()获取自定义属性。

IE 下,even 对象有 x,y 属性,但是没有 pageX,pageY 属性;

Firefox 下,event 对象有 pageX,pageY 属性,但是没有 x,y 属性。

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

Chrome 中文界面下默认会将小于 12px 的文本强制按照 12px 显示,

可通过加入 CSS 属性 -webkit-text-size-adjust: none; 解决。

超链接访问过后 hover 样式就不出现了 被点击访问过的超链接样式不在具有 hover 和 active 了解决方法是改变 CSS 属性的排列顺序:

L-V-H-A : a:link {} a:visited {} a:hover {} a:active {}

19. 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。

20. 为什么要初始化 CSS 样式。

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

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

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

淘宝的样式初始化代码：

body, h1, h2, h3, h4, h5, h6, hr, p, blockquote, dl, dt, dd, ul, ol, li, pre, form, fieldset, legend, button, input, textarea, th, td { margin:0; padding:0; }

body, button, input, select, textarea { font:12px/1.5tahoma, arial, \5b8b\4f53; }

```
h1, h2, h3, h4, h5, h6{ font-size:100%; }

address, cite, dfn, em, var { font-style:normal; }

code, kbd, pre, samp { font-family:couriernew, courier, monospace; }

small{ font-size:12px; }

ul, ol { list-style:none; }

a { text-decoration:none; }

a:hover { text-decoration:underline; }

sup { vertical-align:text-top; }

sub{ vertical-align:text-bottom; }

legend { color:#000; }

fieldset, img { border:0; }

button, input, select, textarea { font-size:100%; }

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

21. absolute 的 containing block(容器块)计算方式跟正常流有什么不同？

无论属于哪种，都要先找到其祖先元素中最近的 position 值不为 static 的元素，然后再判断：

1、若此元素为 inline 元素，则 containing block 为能够包含这个元素生成的第一个和最后一个 inline box 的 padding box (除 margin, border 外的区域) 的最小矩形；

2、否则,则由这个祖先元素的 padding box 构成。

如果都找不到，则为 initial containing block。

补充：

1. static(默认的)/relative：简单说就是它的父元素的内容框（即去掉 padding 的部分）

2. absolute: 向上找最近的定位为 absolute/relative 的元素

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

25.CSS 里的 visibility 属性有个 collapse 属性值是干嘛用的？在不同浏览器下以后什么区别？

其实 visibility 可以有第三种值，就是 collapse。当一个元素的 visibility 属性被设置成 collapse 值后，

对于一般的元素，它的表现跟 hidden 是一样的。但例外的是，如果这个元素是 table 相关的元素，例如 table 行，

table group，table 列，table column group，它的表现却跟 display: none 一样，也就是说，它们占用的空间也会释放。

在谷歌浏览器里，使用 collapse 值和使用 hidden 值没有什么区别

在火狐浏览器、Opera 和 IE11 里，使用 collapse 值的效果就如它的字面意思：table 的行会消失，它的下面一行会补充它的位置。

22.对 BFC 规范(块级格式化上下文：block formatting context)的理解？

（W3C CSS 2.1 规范中的一个概念,它是一个独立容器，决定了元素如何对其内容进行定位,以及与其他元素的关系和相互作用。）

一个页面是由很多个 Box 组成的,元素的类型和 display 属性,决定了这个 Box 的类型。

不同类型的 Box,会参与不同的 Formatting Context（决定如何渲染文档的容器）,

因此 Box 内的元素会以不同的方式渲染,也就是说 BFC 内部的元素和外部的元素不会互相影响。

23.css 定义的权重

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

权重为 1　　div{}

权重为 10　.class1{}

权重为 100　　#id1{}

权重为 100+1=101　　#id1 div{}

权重为 10+1=11　.class1 div{}

权重为 10+10+1=21　.class1 .class2 div{}

如果权重相同，则最后定义的样式会起作用，但是应该避免这种情况出现

24. 请解释一下为什么需要清除浮动？清除浮动的方式

清除浮动是为了清除使用浮动元素产生的影响。浮动的元素，高度会塌陷，而高度的塌陷使我们页面后面的布局不能正常显示

1、父级 div 定义 height；

2、父级 div 也一起浮动；

3、常规的使用一个 class；

```
.clearfix:before, .clearfix:after {
    content: " ";
    display: table;
}
.clearfix:after {
    clear: both;
}
.clearfix {
    *zoom: 1;
}
```

4、SASS 编译的时候，浮动元素的父级 div 定义伪类:after

```
&:after,&:before{
    content: " ";
    visibility: hidden;
```

```
    display: block;
    height: 0;
    clear: both;
}
```

解析原理：

1) display:block 使生成的元素以块级元素显示,占满剩余空间;

2) height:0 避免生成内容破坏原有布局的高度。

3) visibility:hidden 使生成的内容不可见，并允许可能被生成内容盖住的内容可以进行点击和交互;

4）通过 content:"."生成内容作为最后一个元素，至于 content 里面是点还是其他都是可以的，例如 oocss 里面就有经典的 content:".",有些版本可能 content 里面内容为空,一丝冰凉是不推荐这样做的,firefox 直到 7.0 content:”" 仍然会产生额外的空隙；

5）zoom：1 触发 IE hasLayout。

通过分析发现，除了 clear：both 用来闭合浮动的，其他代码无非都是为了隐藏掉 content 生成的内容，

这也就是其他版本的闭合浮动为什么会有 font-size：0，line-height：0。

25. 什么是外边距合并？

外边距合并指的是，当两个垂直外边距相遇时，它们将形成一个外边距。

合并后的外边距的高度等于两个发生合并的外边距的高度中的较大者。

26. zoom:1 的清除浮动原理?

清除浮动，触发 hasLayout；

Zoom 属性是 IE 浏览器的专有属性，它可以设置或检索对象的缩放比例。解决 ie 下比较奇葩的 bug。

譬如外边距（margin）的重叠，浮动清除，触发 ie 的 haslayout 属性等。

来龙去脉大概如下：

当设置了 zoom 的值之后，所设置的元素就会就会扩大或者缩小，高度宽度就会重新计算了，这里一旦改变 zoom 值时其实也会发生重新渲染，运用这个原理，也就解决了 ie 下子元素浮动时候父元素不随着自动扩大的问题。

Zoom 属是 IE 浏览器的专有属性，火狐和老版本的 webkit 核心的浏览器都不支持这个属性。然而，zoom 现在已经被逐步标准化，出现在 CSS 3.0 规范草案中。

目前非 ie 由于不支持这个属性，它们又是通过什么属性来实现元素的缩放呢？

可以通过 css3 里面的动画属性 scale 进行缩放。

27. 使用 CSS 预处理器吗？喜欢那个？（最好用过）

SASS (SASS、LESS 没有本质区别，只因为团队前端都是用的 SASS)

28. CSS 优化、提高性能的方法有哪些？

关键选择器（key selector）。选择器的最后面的部分为关键选择器（即用来匹配目标元素的部分）；

如果规则拥有 ID 选择器作为其关键选择器，则不要为规则增加标签。过滤掉无关的规则（这样样式系统就不会浪费时间去匹配它们了）；

提取项目的通用公有样式，增强可复用性，按模块编写组件；增强项目的协同开发性、可维护性和可扩展性;

使用预处理工具或构建工具（gulp 对 css 进行语法检查、自动补前缀、打包压缩、自动优雅降级）；

29. 浏览器是怎样解析 CSS 选择器的？

样式系统从关键选择器开始匹配，然后左移查找规则选择器的祖先元素。

只要选择器的子树一直在工作，样式系统就会持续左移，直到和规则匹配，或者是因为不匹配而放弃该规则。

30. margin 和 padding 分别适合什么场景使用？

margin 是用来隔开元素与元素的间距；padding 是用来隔开元素与内容的间隔。

margin 用于布局分开元素使元素与元素互不相干；

padding 用于元素与内容之间的间隔，让内容（文字）与（包裹）元素之间有一段

31. ::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。（伪元素由双冒号和伪元素名称组成）

双冒号是在当前规范中引入的，用于区分伪类和伪元素。不过浏览器需要同时支持旧的已经存在的伪元素写法，

比如:first-line、:first-letter、:before、:after 等，

而新的在 CSS3 中引入的伪元素则不允许再支持旧的单冒号的写法。

想让插入的内容出现在其它内容前，使用::before，否者，使用::after；

在代码顺序上，::after 生成的内容也比::before 生成的内容靠后。

如果按堆栈视角，::after 生成的内容会在::before 生成的内容之上

32. 如何修改 chrome 记住密码后自动填充表单的黄色背景 ？

```
input:-webkit-autofill, textarea:-webkit-autofill, select:-webkit-autofill {
    background-color: rgb(250, 255, 189); /* #FAFFBD; */
    background-image: none;
    color: rgb(0, 0, 0);
}
```

33. 设置元素浮动后，该元素的 display 值是多少？

自动变成了 display:block

34. 怎么让 Chrome 支持小于 12px 的文字？

1、用图片：如果是内容固定不变情况下，使用将小于 12px 文字内容切出做图片，这样不影响兼容也不影响美观。

2、使用 12px 及 12px 以上字体大小：为了兼容各大主流浏览器，建议设计美工图时候设置大于或等于 12px 的字体大小，如果是接单的这个时候就需要给客户讲解小于 12px 浏览器不兼容等事宜。

3、继续使用小于 12px 字体大小样式设置：如果不考虑 chrome 可以不用考虑兼容，同时在设置小于 12px 对象设置-webkit-text-size-adjust:none，做到最大兼容考虑。

4、使用 12px 以上字体：为了兼容、为了代码更简单 从新考虑权重下兼容性。

35. 让页面里的字体变清晰，变细用 CSS 怎么做？

-webkit-font-smoothing: antialiased;

36. font-style 属性可以让它赋值为“oblique” oblique 是什么意思？

倾斜的字体样式

37. display:inline-block 什么时候会显示间隙？（携程）

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

38. 设置一个已知 ID 的 div 的 html 内容为 xxxx,字体颜色设置为黑色

方法一：

```
window.onload = function(){
    var box = document.getElementById('box')
    box.innerHTML = 'xxxx'
    var styleNode = document.createElement('style')
    styleNode.innerHTML += '#box{color:block;}'
    document.head.appendChild(styleNode)
}
```

方法二：

```
var dom = document.getElementById(“ID”);
dom.innerHTML = “xxxx”
dom.style.color = “#000”
```

39.你做的页面在哪些流览器测试过？这些浏览器的内核分别是什么？

Trident 内核:IE 系列

Gecko 内核:Firefox

Webkit 内核:Safari

Blink 内核：是基于 Webkit 内核的子项目,使用的浏览器有：

Chrome/opera 等除 IE、Firefox、Safari 之外的几乎所有浏览器

几乎所有国产双内核浏览器（Trident/Blink）如 360、猎豹、qq、百度等

40.每个 HTML 文件里开头都有个很重要的东西,Doctype,知道这是干什么的吗？

文档声明。

<!DOCTYPE> 声明位于文档中的最前面的位置，处于 <html> 标签之前。此标签可告知浏览器文档使用哪种 HTML 或 XHTML 规范。（重点：告诉浏览器按照何种规范解析页面）

IE 下如不书写文档声明会使用怪异模式解析网页导致一系列 CSS 兼容性问题。

41.div+css 的布局较 table 布局有什么优点？

正常场景一般都适用 div+CSS 布局，优点：

结构与样式分离

代码语义性好

更符合 HTML 标准规范

SEO 友好

Table 布局的适用场景：

某种原因不方便加载外部 CSS 的场景，例如邮件正文，此时用 table 布局可以在无 css 情况下保持页面布局正常。

42.img 的 alt 与 title 有何异同？ strong 与 em 的异同？

a:alt(alt text):为不能显示图像、窗体或 applets 的用户代理（UA），alt 属性用来指定替换文字。替换文字的语言由 lang 属性指定。(在 IE 浏览器下会在没有 title 时把 alt 当成 tool tip 显示)

title(tool tip):该属性为设置该属性的元素提供建议性的信息。

em:表现为斜体，语义表示强调

strong:表现为粗体，语义为强烈语气，强调程度超过 em

43. 你能描述一下渐进增强和优雅降级之间的不同吗?

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：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，而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。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。

“优雅降级”观点：

“优雅降级”观点认为应该针对那些最高级、最完善的浏览器来设计网站。而将那些被认为“过时”或有功能缺失的浏览器下的测试工作安排在开发周期的最后阶段，并把测试对象限定为主流浏览器（如 IE、Mozilla 等）的前一个版本。

在这种设计范例下，旧版的浏览器被认为仅能提供“简陋却无妨 (poor, but passable)”的浏览体验。你可以做一些小的调整来适应某个特定的浏览器。但由于它们并非我们所关注的焦点，因此除了修复较大的错误之外，其它的差异将被直接忽略。

“渐进增强”观点

“渐进增强”观点则认为应关注于内容本身。

内容是我们建立网站的诱因。有的网站展示它，有的则收集它，有的寻求，有的操作，还有的网站甚至会包含以上的种种，但相同点是它们全都涉及到内容。这使得“渐进增强”成为一种更为合理的设计范例。这也是它立即被 Yahoo! 所采纳并用以构建其“分级式浏览器支持 (Graded Browser Support)”策略的原因所在。

44. 为什么利用多个域名来存储网站资源会更有效？

CDN 缓存更方便

突破浏览器并发限制

节约 cookie 带宽

节约主域名的连接数，优化页面响应速度

防止不必要的安全问题

45. 请谈一下你对网页标准和标准制定机构重要性的理解。

网页标准和标准制定机构都是为了能让 web 发展的更‘健康’，开发者遵循统一的标准，降低开发难度，开发成本，SEO 也会更好做，也不会因为滥用代码导致各种 BUG、安全问题，最终提高网站易用性。

46. 请描述一下 cookies，sessionStorage 和 localStorage 的区别？

sessionStorage 用于本地存储一个会话（session）中的数据，这些数据只有在同一个会话中的页面才能访问并且当会话结束后数据也随之销毁。因此 sessionStorage 不是一种持久化的本地存储，仅仅是会话级别的存储。而 localStorage 用于持久化的本地存储，除非主动删除数据，否则数据是永远不会过期的。

web storage 和 cookie 的区别：

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

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

47. 简述一下 src 与 href 的区别。

src 用于替换当前元素，href 用于在当前文档和引用资源之间确立联系。

src 是 source 的缩写，指向外部资源的位置，指向的内容将会嵌入到文档中当前标签所在位置；在请求 src 资源时会将其指向的资源下载并应用到文档内，例如 js 脚本，img 图片和 frame 等元素。

<script src =”js.js”></script>

当浏览器解析到该元素时，会暂停其他资源的下载和处理，直到将该资源加载、编译、执行完毕，图片和框架等元素也如此，类似于将所指向资源嵌入当前标签内。这也是为什么将 js 脚本放在底部而不是头部。

href 是 Hypertext Reference 的缩写，指向网络资源所在位置，建立和当前元素（锚点）或当前文档（链接）之间的链接，如果我们在文档中添加

<link href=”common.css” rel=”stylesheet”/>

那么浏览器会识别该文档为 css 文件，就会并行下载资源并且不会停止对当前文档的处理。这也是为什么建议使用 link 方式来加载 css，而不是使用@import 方式。

48. 知道的网页制作会用到的图片格式有哪些？

png-8，png-24，jpeg，gif，svg。

但是上面的那些都不是面试官想要的最后答案。面试官希望听到是 Webp。（是否有关注新技术，新鲜事物）

科普一下 Webp：WebP 格式，谷歌（google）开发的一种旨在加快图片加载速度的图片格式。图片压缩体积大约只有 JPEG 的 2/3，并能节省大量的服务器带宽资源和数据空间。Facebook Ebay 等知名网站已经开始测试并使用 WebP 格式。

在质量相同的情况下，WebP 格式图像的体积要比 JPEG 格式图像小 40%

49. 在 css/js 代码上线之后开发人员经常会优化性能，从用户刷新网页开始，一次 js 请求一般情况下有哪些地方会有缓存处理？

答案：dns 缓存，cdn 缓存，浏览器缓存，服务器缓存。

50. 一个页面上有大量的图片（大型电商网站），加载很慢，你有哪些方法优化这些图片的加载，给用户更好的体验。

图片懒加载，在页面上的未可视区域可以添加一个滚动条事件，判断图片位置与浏览器顶端的距离与页面的距离，如果前者小于后者，优先加载。

如果为幻灯片、相册等，可以使用图片预加载技术，将当前展示图片的前一张和后一张优先下载。

如果图片为 css 图片，可以使用 CSSsprite，SVGsprite，Iconfont、Base64 等技术。

如果图片过大，可以使用特殊编码的图片，加载时会先加载一张压缩的特别厉害的缩略图，以提高用户体验。

如果图片展示区域小于图片的真实大小，则因在服务器端根据业务需要先行进行图片压缩，图片压缩后大小与展示一致。

51. 你如何理解 HTML 结构的语义化？

HTML 结构语义化：

更符合 W3C 统一的规范标准，是技术趋势。

没有样式时浏览器的默认样式也能让页面结构很清晰。

对功能障碍用户友好。屏幕阅读器（如果访客有视障）会完全根据你的标记来“读”你的网页。

对其他非主流终端设备友好。例如机顶盒、PDA、各种移动终端。

对 SEO 友好。

52. 谈谈以前端角度出发做好 SEO 需要考虑什么？

搜索引擎主要以:

外链数量和质量

网页内容和结构

来决定某关键字下的网页搜索排名。

前端应该注意网页结构和内容方面的情况：

Meta 标签优化

主要包括主题（Title)，网站描述(Description)。还有一些其它的隐藏文字比如 Author（作者），Category（目录），Language（编码语种）等。

符合 W3C 规范的语义性标签的使用。

如何选取关键词并在网页中放置关键词

搜索就得用关键词。关键词分析和选择是 SEO 最重要的工作之一。首先要给网站确定主关键词（一般在 5 个上下），然后针对这些关键词进行优化，包括关键词密度（Density），相关度（Relavancy），突出性（Prominency）等等。

53.有哪项方式可以对一个 DOM 设置它的 CSS 样式？

外部样式表，引入一个外部 css 文件

内部样式表，将 css 代码放在 <head> 标签内部

内联样式，将 css 样式直接定义在 HTML 元素内部

54.CSS 都有哪些选择器？

一、基本选择器

1. * 通用元素选择器，匹配任何元素
2. E 标签选择器，匹配所有使用 E 标签的元素
3. .infoclass 选择器，匹配所有 class 属性中包含 info 的元素
4. #footer id 选择器，匹配所有 id 属性等于 footer 的元素

二、多元素的组合选择器

5. E,F 多元素选择器，同时匹配所有 E 元素或 F 元素，E 和 F 之间用逗号分隔
6. E F 后代元素选择器，匹配所有属于 E 元素后代的 F 元素，E 和 F 之间用空格分隔
7. E > F子元素选择器，匹配所有 E 元素的子元素 F
8. E + F毗邻元素选择器，匹配所有紧随 E 元素之后的同级元素 F

三、CSS 2.1 属性选择器

9. E[att] 匹配所有具有 att 属性的 E 元素，不考虑它的值。（注意：E 在此处可以省略，比如"[cheacked]"。以下同。）
10. E[att=val] 匹配所有 att 属性等于"val"的 E 元素
11. E[att~=val] 匹配所有 att 属性具有多个空格分隔的值、其中一个值等于"val"的 E 元素
12. E[att|=val] 匹配所有 att 属性具有多个连字号分隔（hyphen-separated）的值、其中一个值以"val"开头的 E 元素，主要用于 lang 属性，比如"en"、"en-us"、"en-gb"等等

四、CSS 2.1 中的伪类

13. E:first-child 匹配父元素的第一个子元素

14. E:link 匹配所有未被点击的链接
15. E:visited 匹配所有已被点击的链接
16. E:active 匹配鼠标已经其上按下、还没有释放的 E 元素
17. E:hover 匹配鼠标悬停其上的 E 元素
18. E:focus 匹配获得当前焦点的 E 元素
19. E:lang(c) 匹配 lang 属性等于 c 的 E 元素

五、 CSS 2.1 中的伪元素

20. E:first-line 匹配 E 元素的第一行
21. E:first-letter 匹配 E 元素的第一个字母
22. E:before 在 E 元素之前插入生成的内容
23. E:after 在 E 元素之后插入生成的内容

六、CSS 3 的同级元素通用选择器

24. E ~ F匹配任何在 E 元素之后的同级 F 元素

七、CSS 3 属性选择器

25. E[att^="val"] 属性 att 的值以"val"开头的元素
26. E[att$="val"] 属性 att 的值以"val"结尾的元素
27. E[att*="val"] 属性 att 的值包含"val"字符串的元素

八、CSS 3 中与用户界面有关的伪类

28. E:enabled 匹配表单中激活的元素
29. E:disabled 匹配表单中禁用的元素
30. E:checked 匹配表单中被选中的 radio（单选框）或 checkbox（复选框）元素
31. E::selection 匹配用户当前选中的元素

九、CSS 3 中的结构性伪类

32. E:root 匹配文档的根元素，对于 HTML 文档，就是 HTML 元素

33. E:nth-child(n) 匹配其父元素的第 n 个子元素，第一个编号为 1

34. E:nth-last-child(n) 匹配其父元素的倒数第 n 个子元素，第一个编号为 1

35. E:nth-of-type(n) 与:nth-child()作用类似，但是仅匹配使用同种标签的元素

36. E:nth-last-of-type(n) 与:nth-last-child() 作用类似，但是仅匹配使用同种标签的元素

37. E:last-child 匹配父元素的最后一个子元素，等同于:nth-last-child(1)

38. E:first-of-type 匹配父元素下使用同种标签的第一个子元素，等同于:nth-of-type(1)

39. E:last-of-type 匹配父元素下使用同种标签的最后一个子元素，等同于:nth-last-of-type(1)

40. E:only-child 匹配父元素下仅有的一个子元素，等同于:first-child:last-child 或 :nth-child(1):nth-last-child(1)

41. E:only-of-type 匹配父元素下使用同种标签的唯一一个子元素，等同于:first-of-type:last-of-type 或 :nth-of-type(1):nth-last-of-type(1)

42. E:empty 匹配一个不包含任何子元素的元素，注意，文本节点也被看作子元素

十、CSS 3 的反选伪类

43. E:not(s) 匹配不符合当前选择器的任何元素

十一、CSS 3 中的 :target 伪类

44. E:target 匹配文档中特定"id"点击后的效果

55. CSS 中可以通过哪些属性定义，使得一个 DOM 元素不显示在浏览器可视范围内？

设置 display 属性为 none，或者设置 visibility 属性为 hidden

设置宽高为 0，设置透明度为 0，设置 z-index 位置在-1000em

设置 text-indent:-9999px;

56. 超链接访问过后 hover 样式就不出现的问题是什么？如何解决？

答案：被点击访问过的超链接样式不再具有 hover 和 active 了,解决方法是改变 CSS 属性的排列顺序: L-V-H-A（link,visited,hover,active）

57. 什么是 Css Hack？ie6,7,8 的 hack 分别是什么？

答案：针对不同的浏览器写不同的 CSS code 的过程，就是 CSS hack。

示例如下：

```
#test{
 background-color:yellow;    /*ie8*/
 +background-color:pink;      /*ie7*/
 _background-color:orange;     /*ie6*/   }
```

更好的方式是使用 IE 条件判断语句：

<! – [if lte IE 6]>

内容

<![endif] – >

等

58. 行内元素和块级元素的具体区别是什么？行内元素的 padding 和 margin 可设置吗？

块级元素(block)特性：

总是独占一行，表现为另起一行开始，而且其后的元素也必须另起一行显示;

宽度(width)、高度(height)、内边距(padding)、边框(border)和外边距(margin)都可控制;

内联元素(inline)特性：

和相邻的内联元素在同一行;

宽度(width)、高度(height)、内边距的 top/bottom(padding-top/padding-bottom)和外边距的 top/bottom(margin-top/margin-bottom)还有 border top/bottom 都不可改变（也就是 padding 和 margin 的 left 和 right 是可以设置的），就是里面文字或图片的大小。

浏览器还有默认的天生 inline-block 元素（拥有内在尺寸，可设置高宽，但不会自动换行）

答案：<input> 、<img> 、<button> 、<texterea> 、<label>。

59. 什么是外边距重叠？重叠的结果是什么？

外边距重叠就是 margin-collapse。

在 CSS 当中，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，并且因而所结合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。

60. rgba()和 opacity 的透明效果有什么不同？

rgba()和 opacity 都能实现透明效果，但最大的不同是 opacity 作用于元素，以及元素内的所有内容的透明度，而 rgba()只作用于元素的颜色或其背景色。（设置 rgba 透明的元素的子元素不会继承透明效果！）

61. css 中可以让文字在垂直和水平方向上重叠的两个属性是什么？

垂直方向：line-height

水平方向：letter-spacing

那么问题来了，关于 letter-spacing 的妙用知道有哪些么？

答案:可以用于消除 inline-block 元素间的换行符空格间隙问题。

62 如何垂直居中一个浮动元素？

```
// 方法一：已知元素的高宽

#div1{

  background-color:#6699FF;

  width:200px;

  height:200px;

  position: absolute;       /*父元素需要相对定位*/
```

```
top: 50%;
left: 50%;
margin-top:-100px ;  /*二分之一的 height，width*/
margin-left: -100px;
}
```

```
//方法二:未知元素的高宽
#div1{
width: 200px;
height: 200px;
background-color: #6699FF;

margin:auto;
position: absolute;      /*父元素需要相对定位*/
left: 0;
top: 0;
right: 0;
bottom: 0;
}
```

63.如何垂直居中一个<img>?

（用更简便的方法。）

```
/*<img>的容器设置如下*/
#container {
    display:table-cell;
    text-align:center;
    vertical-align:middle;}
```

64.px 和 em 的区别。

px 和 em 都是长度单位，区别是：

px 值固定，容易计算。

em 值不固定，是相对单位，其相对应父级元素的字体大小会调整

65. Sass、LESS 是什么？大家为什么要使用他们？

他们是 CSS 预处理器。他是 CSS 上的一种抽象层。他们是一种特殊的语法/语言编译成 CSS。

例如 Less 是一种动态样式语言. 将 CSS 赋予了动态语言的特性，如变量，继承，运算，函数. LESS 既可以在客户端上运行 (支持 IE 6+, Webkit, Firefox)，也可一在服务端运行 (借助 Node.js)。

为什么要使用它们？

结构清晰，便于扩展。

可以方便地屏蔽浏览器私有语法差异。这个不用多说，封装对浏览器语法差异的重复处理，减少无意义的机械劳动。

可以轻松实现多重继承。

完全兼容 CSS 代码，可以方便地应用到老项目中。LESS 只是在 CSS 语法上做了扩展，所以老的 CSS 代码也可以与 LESS 代码一同编译。

66. display:none 与 visibility:hidden 的区别是什么？

display : 隐藏对应的元素但不挤占该元素原来的空间。

visibility: 隐藏对应的元素并且挤占该元素原来的空间。

即是，使用 CSS display:none 属性后，HTML 元素（对象）的宽度、高度等各种属性值都将“丢失”;而使用 visibility:hidden 属性后，HTML 元素（对象）仅仅是在视觉上看不见（完全透明），而它所占据的空间位置仍然存在。

67. 简介盒子模型：

CSS 的盒子模型有两种：IE 盒子模型、标准的 W3C 盒子模型模型

盒模型：内容、内边距、外边距（一般不计入盒子实际宽度）、边框

68. 为什么要初始化样式？

用于浏览器默认 css 样式的存在并且不同浏览器对相同 HTML 标签的默认样式不同，若不初始化会造成不同浏览器之间的显示差异。

69. BFC 是什么？

BFC 就是“块级格式化上下文”的意思，创建了 BFC 的元素就是一个独立的盒子，不过只有 Block-level box 可以参与创建 BFC， 它规定了内部的 Block-level Box 如何布局，并且与这个独立盒子里的布局不受外部影响，当然它也不会影响到外面的元素。

BFC 有一下特性：

内部的 Box 会在垂直方向，从顶部开始一个接一个地放置。

Box 垂直方向的距离由 margin 决定。属于同一个 BFC 的两个相邻 Box 的 margin 会发生叠加

每个元素的 margin box 的左边， 与包含块 border box 的左边相接触(对于从左往右的格式化，否则相反)。即使存在浮动也是如此。

BFC 的区域不会与 float box 叠加。

BFC 就是页面上的一个隔离的独立容器，容器里面的子元素不会影响到外面的元素，反之亦然。

计算 BFC 的高度时，浮动元素也参与计算。

如何触发 BFC?

float 除了 none 以外的值

overflow 除了 visible 以外的值（hidden，auto，scroll ）

display (table-cell，table-caption，inline-block, flex, inline-flex)

position 值为（absolute，fixed）

fieldset 元素

70. Doctype 的作用？严格模式与混杂模式的区别？

<!DOCTYPE>,文档声明；用于告知浏览器该以何种模式来渲染文档

严格模式下：页面排版及 JS 解析是以该浏览器支持的最高标准来执行

混杂模式：不严格按照标准执行，主要用来兼容旧的浏览器，向后兼容

71. IE 的双边距 BUG：块级元素 float 后设置横向 margin，ie6 显示的 margin 比设置的较大。

解决：加入_display：inline

72.HTML 与 XHTML——二者有什么区别？

1. 所有的标记都必须要有一个相应的结束标记

2. 所有标签的元素和属性的名字都必须使用小写

3. 所有的 XML 标记都必须合理嵌套

4. 所有的属性必须用引号 "" 括起来

5. 把所有 < 和 & 特殊符号用编码表示

6. 给所有属性赋一个值

7. 不要在注释内容中使用 "--"

8. 图片必须有说明文字

73.html 常见兼容性问题？

1.双边距 BUG float 引起的 使用 display

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

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

4.IE z-index 问题 给父级添加 position:relative

5.Png 透明 使用 js 代码 改

6.Min-height 最小高度 ！ Important 解决’

7.select 在 ie6 下遮盖 使用 iframe 嵌套

8.为什么没有办法定义 1px 左右的宽度容器（IE6 默认的行高造成的，使用 over:hidden,zoom:0.08 line-height:1px）

9.IE5-8 不支持 opacity，解决办法：

```
.opacity {
    opacity: 0.4
    filter: alpha(opacity=60); /* for IE5-7 */
    -ms-filter: "progid:DXImageTransform.Microsoft.Alpha(Opacity=60)"; /* for IE 8*/
}
```

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

74. 对 WEB 标准以及 W3C 的理解与认识

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

75. 行内元素有哪些?块级元素有哪些?

答：块级元素：div p h1 h2 h3 h4 form ul

行内元素: a b br i span input select

76. 前端页面有哪三层构成，分别是什么?作用是什么?

答：结构层 Html 表示层 CSS 行为层 js。

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

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

（2）、严格模式的排版和 JS 运作模式是,以该浏览器支持的最高标准运行。

（3）、在混杂模式中，页面以宽松的向后兼容的方式显示。模拟老式浏览器的行为以防止站点无法工作。

（4）、DOCTYPE 不存在或格式不正确会导致文档以混杂模式呈现。

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

1. block 象块类型元素一样显示。

none 缺省值。向行内元素类型一样显示。

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

list-item 象块类型元素一样显示，并添加样式列表标记。

2. position 的值

*absolute

生成绝对定位的元素，相对于 static 定位以外的第一个父元素进行定位。

*fixed （老 IE 不支持）

生成绝对定位的元素，相对于浏览器窗口进行定位。

* relative

生成相对定位的元素，相对于其正常位置进行定位。

* static 默认值。没有定位，元素出现在正常的流中

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

* inherit 规定从父元素继承 position 属性的值。

79.对 WEB 标准以及 W3C 的理解与认识

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

80.css 的基本语句构成是?

选择器{属性 1:值 1;属性 2:值 2;……}

81.CSS 中可以通过哪些属性定义，使得一个 DOM 元素不显示在浏览器可视范围内？

最基本的：

设置 display 属性为 none，或者设置 visibility 属性为 hidden

技巧性：

设置宽高为 0，设置透明度为 0，设置 z-index 位置在-1000

82.行内元素和块级元素的具体区别是什么？行内元素的 padding 和 margin 可设置吗？

块级元素(block)特性：

总是独占一行，表现为另起一行开始，而且其后的元素也必须另起一行显示;

宽度(width)、高度(height)、内边距(padding)和外边距(margin)都可控制;

内联元素(inline)特性：

和相邻的内联元素在同一行;

宽度(width)、高度(height)、内边距的 top/bottom(padding-top/padding-bottom)和外边距的 top/bottom(margin-top/margin-bottom)都不可改变（也就是 padding 和 margin 的 left 和 right 是可以设置的），就是里面文字或图片的大小。

那么问题来了，浏览器还有默认的天生 inline-block 元素（拥有内在尺寸，可设置高宽，但不会自动换行），有哪些？

答案：<input> 、<img> 、<button> 、<textarea> 、<label>

83.什么是外边距重叠？重叠的结果是什么？

外边距重叠就是 margin-collapse。

在 CSS 当中，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，并且因而所结合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。

84.说 display 属性有哪些？可以做什么？

display:block 行内元素转换为块级元素

display:inline 块级元素转换为行内元素

display:inline-block 转为内联元素

85.哪些 css 属性可以继承？

可继承： font-size font-family color

不可继承 ：border padding margin width height ;

86.css 优先级算法如何计算？

!important > id > class > 标签

!important 比 内联优先级高

*优先级就近原则，样式定义最近者为准;

*以最后载入的样式为准;

87.b 标签和 strong 标签,i 标签和 em 标签的区别？

后者有语义，前者则无。

88.有那些行内元素、有哪些块级元素、盒模型？

1.内联元素(inline element)

a – 锚点

abbr – 缩写

acronym – 首字

b – 粗体(不推荐)

big – 大字体

br – 换行

em – 强调

font – 字体设定(不推荐)

i – 斜体

img – 图片

input – 输入框

label – 表格标签

s – 中划线(不推荐)

select – 项目选择

small – 小字体文本

span – 常用内联容器，定义文本内区块

strike – 中划线

strong – 粗体强调

sub – 下标

sup – 上标

textarea – 多行文本输入框

tt – 电传文本

u – 下划线

var – 定义变量

2、块级元素

address – 地址

blockquote – 块引用

center – 举中对齐块

dir – 目录列表

div – 常用块级容易，也是 css layout 的主要标签

dl – 定义列表

fieldset – form 控制组

form – 交互表单

h1 – 大标题

h2 – 副标题

h3 – 3 级标题

h4 – 4 级标题

h5 – 5 级标题

h6 – 6 级标题

hr – 水平分隔线

isindex – input prompt

menu – 菜单列表

noframes – frames 可选内容，（对于不支持 frame 的浏览器显示此区块内容）

noscript – ）可选脚本内容（对于不支持 script 的浏览器显示此内容）

ol – 排序表单

p – 段落

pre – 格式化文本

table – 表格

ul – 非排序列表

3.CSS 盒子模型包含四个部分组成：

内容、填充（padding）、边框（border）、外边界（margin）

89.有哪些选择符，优先级的计算公式是什么？行内样式和！important 哪个优先级高？

#ID > .class > 标签选择符 !important 优先级高

90. 我想让行内元素跟上面的元素距离 10px，加 margin-top 和 padding-top 可以吗？

margin-top,padding-top 无效

二、HTML5+CSS3

1. CSS3 有哪些新特性？

1. CSS3 实现圆角（border-radius），阴影（box-shadow），

2. 对文字加特效（text-shadow、），线性渐变（gradient），旋转（transform）

3.transform:rotate(9deg) scale(0.85,0.90) translate(0px,-30px) skew(-9deg,0deg);// 旋转,缩放,定位,倾斜

4. 增加了更多的 CSS 选择器　多背景 rgba

5. 在 CSS3 中唯一引入的伪元素是 ::selection.

6. 媒体查询，多栏布局

7. border-image

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

新特性：

1. 拖拽释放(Drag and drop) API

2. 语义化更好的内容标签（header,nav,footer,aside,article,section）

3. 音频、视频 API(audio,video)

4. 画布(Canvas) API

5. 地理(Geolocation) API

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

7. sessionStorage 的数据在浏览器关闭后自动删除

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

9. 新的技术 webworker, websocket, Geolocation

移除的元素：

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

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

支持 HTML5 新标签：

1. IE8/IE7/IE6 支持通过 document.createElement 方法产生的标签，可以利用这一特性让这些浏览器支持 HTML5 新标签，浏览器支持新标签后，还需要添加标签默认的样式（当然最好的方式是直接使用成熟的框架、使用最多的是 html5shim 框架）：

<!--[if lt IE 9]>

<script> src="http://html5shim.googlecode.com/svn/trunk/html5.js"</script>

<![endif]-->

如何区分：

DOCTYPE 声明新增的结构元素、功能元素

3. 本地存储（Local Storage ）和 cookies（储存在用户本地终端上的数据）之间的区别是什么？

Cookies:服务器和客户端都可以访问；大小只有 4KB 左右；有有效期，过期后将会删除；

本地存储：只有本地浏览器端可访问数据，服务器不能访问本地存储直到故意通过 POST 或者 GET 的通道发送到服务器；每个域 5MB；没有过期数据，它将保留知道用户从浏览器清除或者使用 Javascript 代码移除

4. 如何实现浏览器内多个标签页之间的通信?

调用 localstorge、cookies 等本地存储方式

5. 你如何对网站的文件和资源进行优化？

文件合并

文件最小化/文件压缩

使用 CDN 托管

缓存的使用

6. 什么是响应式设计？

低成本实现一套代码一个网页在多终端多设备下访问达到一定用户体验的开发方式。其布局会根据终端情况自适应调整达到一定水平的用户体验。

7.新的 HTML5 文档类型和字符集是？

HTML5 文档类型：<!doctype html>

HTML5 使用的编码<meta charset=”UTF-8”>

8.HTML5 Canvas 元素有什么用？

Canvas 元素用于在网页上绘制图形，该元素标签强大之处在于可以直接在 HTML 上进行图形操作。

9.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。

: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.如何在 HTML5 页面中嵌入音频?

HTML 5 包含嵌入音频文件的标准方式，支持的格式包括 MP3、Wav 和 Ogg：

```
<audio controls>
 <source src="jamshed.mp3" type="audio/mpeg">
  Your browser does'nt support audio embedding feature.
</audio>
```

11.描述一段语义的 html 代码吧。

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

就是基于语义化设计原则）

```
< div id="header">
< h1>标题< /h1>
< h2>专注 Web 前端技术< /h2>
```

< /div>

语义 HTML 具有以下特性：

文字包裹在元素中，用以反映内容。例如：

段落包含在 元素中。

顺序表包含在<ol>元素中。

从其他来源引用的大型文字块包含在<blockquote>元素中。

HTML 元素不能用作语义用途以外的其他目的。例如：

<h1>包含标题，但并非用于放大文本。

<blockquote>包含大段引述，但并非用于文本缩进。

空白段落元素（ ）并非用于跳行。

文本并不直接包含任何样式信息。例如：

不使用 <font> 或 等格式标记。

类或 ID 中不引用颜色或位置。

12. HTML5 的离线储存？

localStorage 长期存储数据，浏览器关闭后数据不丢失；

sessionStorage 数据在浏览器关闭后自动删除。

13. 自己对标签语义化的理解

网页使用什么 HTML 标签要看这个元素是什么元素，而不是看这个元素像什么元素。

例如我们用 h2 标签，是因为这个元素是二级标题 ，而不是因为它看起来比较字体比较粗比较大。

14. HTML5 为什么只需要写 <!DOCTYPE HTML>？

HTML5 不基于 SGML，因此不需要对 DTD 进行引用，但是需要 doctype 来规范浏览器的行为（让浏览器按照它们应该的方式来运行）；

而 HTML4.01 基于 SGML,所以需要对 DTD 进行引用，才能告知浏览器文档所使用的文档类型。

15. HTML5 的离线储存怎么使用，工作原理能不能解释一下？

在用户没有与因特网连接时，可以正常访问站点或应用，在用户与因特网连接时，更新用户机器上的缓存文件。

原理：HTML5 的离线存储是基于一个新建的.appcache 文件的缓存机制(不是存储技术)，通过这个文件上的解析清单离线存储资源，这些资源就会像 cookie 一样被存储了下来。之后当网络在处于离线状态下时，浏览器会通过被离线存储的数据进行页面展示。

如何使用：

1、页面头部像下面一样加入一个 manifest 的属性；

2、在 cache.manifest 文件的编写离线存储的资源；

```
CACHE MANIFEST

#v0.11

CACHE:

js/app.js

css/style.css

NETWORK:

resourse/logo.png

FALLBACK:

/ /offline.html
```

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

17. 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。

18. 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

19. 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。

20. 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。

21. 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；

22. 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

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

一个用于页面布局的全新 CSS3 功能，Flexbox 可以把列表放在同一个方向（从上到下排列，从左到右），并让列表能延伸到占用可用的空间。

较为复杂的布局还可以通过嵌套一个伸缩容器（flex container）来实现。

采用 Flex 布局的元素，称为 Flex 容器（flex container），简称"容器"。

它的所有子元素自动成为容器成员，称为 Flex 项目（flex item），简称"项目"。

常规布局是基于块和内联流方向，而 Flex 布局是基于 flex-flow 流可以很方便的用来做局中，能对不同屏幕大小自适应。

在布局上有了比以前更加灵活的空间。

26. 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;
}
```

27. 移动端的布局用过媒体查询吗？

假设你现在正用一台显示设备来阅读这篇文章，同时你也想把它投影到屏幕上，或者打印出来，

而显示设备、屏幕投影和打印等这些媒介都有自己的特点，CSS 就是为文档提供在不同媒介上展示的适配方法

当媒体查询为真时，相关的样式表或样式规则会按照正常的级联规被应用。 当媒体查询返回假，

标签上带有媒体查询的样式表 仍将被下载 （只不过不会被应用）。

包含了一个媒体类型和至少一个使用 宽度、高度和颜色等媒体属性来限制样式表范围的表达式。

CSS3 加入的媒体查询使得无需修改内容便可以使样式应用于某些特定的设备范围。

@media (min-width: 700px) and (orientation: landscape){ .sidebar { display: none; } }

28. position:fixed;在 android 下无效怎么处理？

fixed 的元素是相对整个页面固定位置的，你在屏幕上滑动只是在移动这个所谓的 viewport，

原来的网页还好好的在那，fixed 的内容也没有变过位置，

所以说并不是 iOS 不支持 fixed，只是 fixed 的元素不是相对手机屏幕固定的。

<meta name="viewport" content="width=device-width, initial-scale=1.0,

maximum-scale=1.0, minimum-scale=1.0, user-scalable=no"/>

29. 什么是 Cookie 隔离？

（或者说：请求资源的时候不要让它带 cookie 怎么做）

如果静态文件都放在主域名下，那静态文件请求的时候都带有的 cookie 的数据提交给 server 的，非常浪费流量,所以不如隔离开。

因为 cookie 有域的限制，因此不能跨域提交请求，故使用非主要域名的时候，请求头中就不会带有 cookie 数据，

这样可以降低请求头的大小，降低请求时间，从而达到降低整体请求延时的目的。

同时这种方式不会将 cookie 传入 Web Server，也减少了 Web Server 对 cookie 的处理分析环节，

提高了 webserver 的 http 请求的解析速度。

30. 什么是 CSS 预处理器 / 后处理器？

预处理器例如：LESS、Sass、Stylus，用来预编译 Sass 或 less，增强了 css 代码的复用性，

还有层级、mixin、变量、循环、函数等，具有很方便的 UI 组件模块化开发能力，极大的提高工作效率。

后处理器例如：PostCSS，通常被视为在完成的样式表中根据 CSS 规范处理 CSS，让其更有效；目前最常做的

是给 CSS 属性添加浏览器私有前缀，实现跨浏览器兼容性的问题。

31. 如何隐藏一个 DOM 元素

设置 css display 属性为 none,效果：元素不显示，不占位

设置 css visibility 属性为 hidden,效果：元素不显示，但占位

opacity 值为 0

position 值为 absolute ，并且将其移到不可见区域

HTML5 元素的属性值 hidden,（就是给元素声明一个 hidden 属性值）,<div hidden>

元素的 font-size 、 line-height 、 width 和 height 设置为 0

设置元素的 transform 的 translateX 或者 translateY 的值为 -100%

设置元素的 transform 的 transform: scale(0);

三、JS 基础

1. 介绍 js 的基本数据类型。

Undefined、Null、Boolean、Number、String、

ECMAScript 2015 新增:Symbol(创建后独一无二且不可变的数据类型)

2. 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

3. 说几条写 JavaScript 的基本规范？

1.不要在同一行声明多个变量。

2.请使用 ===/!==来比较 true/false 或者数值

3.使用对象字面量替代 new Array 这种形式

4.不要使用全局函数。

5.Switch 语句必须带有 default 分支

6.函数不应该有时候有返回值，有时候没有返回值。

7.For 循环必须使用大括号

8.If 语句必须使用大括号

9.for-in 循环中的变量 应该使用 var 关键字明确限定作用域，从而避免作用域污染。

4. JavaScript 原型，原型链 ？有什么特点？

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

如果这个对象内部不存在这个属性，那么他就会去 prototype 里找这个属性，这个 prototype 又会有自己的 prototype，

于是就这样一直找下去，也就是我们平时所说的原型链的概念。

关系：instance.constructor.prototype = instance.__proto__

特点：

JavaScript 对象是通过引用来传递的，我们创建的每个新对象实体中并没有一份属于自己的原型副本。

当我们修改原型时，与之相关的对象也会继承这一改变。

当我们需要一个属性的时，Javascript 引擎会先看当前对象中是否有这个属性， 如果没有的话，

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

```
function Func(){}

Func.prototype.name = "Sean";

Func.prototype.getInfo = function() {

  return this.name;

}
```

```
var person = new Func();//现在可以参考 var person = Object.create(oldObject);
console.log(person.getInfo());//它拥有了 Func 的属性和方法
//"Sean"
console.log(Func.prototype);
// Func { name="Sean", getInfo=function()}
```

5. JavaScript 有几种类型的值？你能画一下他们的内存图吗？

栈：原始数据类型（Undefined，Null，Boolean，Number、String）

堆：引用数据类型（对象、数组和函数）

两种类型的区别是：存储位置不同；

原始数据类型直接存储在栈(stack)中的简单数据段，占据空间小、大小固定，属于被频繁使用数据，所以放入栈中存储；

引用数据类型存储在堆(heap)中的对象,占据空间大、大小不固定。如果存储在栈中，将会影响程序运行的性能；

引用数据类型在栈中存储了指针，该指针指向堆中该实体的起始地址。当解释器寻找引用值时，

会首先检索其在栈中的地址，取得地址后从堆中获得实体

6. 如何将字符串转化为数字，例如'12.3b'？

```
parseFloat('12.3b');
```

正则表达式，'12.3b'.match(/(\d)+(\.)?(\d)+/g)[0] * 1，但是这个不太靠谱，提供一种思路而已。

7. 如何将浮点数点左边的数每三位添加一个逗号，如 12000000.11 转化为『12,000,000.11』？

```
function commafy(num){
    return num && num
        .toString()
        .replace(/(\d)(?=(\d{3})+\.)/g, function($1, $2){
            return $2 + ',';
        });
}
```

8. 如何实现数组的随机排序？

方法一：javascript

```
var arr = [1,2,3,4,5,6,7,8,9,10];
function randSort1(arr){
    for(var i = 0,len = arr.length;i < len; i++ ){
        var rand = parseInt(Math.random()*len);
        var temp = arr[rand];
        arr[rand] = arr[i];
        arr[i] = temp;
    }
    return arr;
}
console.log(randSort1(arr));
```

方法二：

```
var arr = [1,2,3,4,5,6,7,8,9,10];
function randSort2(arr){
    var mixedArray = [];
    while(arr.length > 0){
        var randomIndex = parseInt(Math.random()*arr.length);
        mixedArray.push(arr[randomIndex]);
        arr.splice(randomIndex, 1);
    }
    return mixedArray;
}
console.log(randSort2(arr));
```

方法三：

```
var arr = [1,2,3,4,5,6,7,8,9,10];
arr.sort(function(){
```

```
        return Math.random() - 0.5;
    })
    console.log(arr);
```

9. javascript 创建对象的几种方式？（腾讯）

javascript 创建对象简单的说,无非就是使用内置对象或各种自定义对象，当然还可以用 JSON；但写法有很多种，也能混合使用。

1、对象字面量的方式

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

2、用 function 来模拟无参的构造函数

```
function Person(){}
var person=new Person();//定义一个 function，如果使用 new"实例化",该 function 可以看作是一个 Class
person.name="Mark";
person.age="25";
person.work=function(){
alert(person.name+" hello...");
}
person.work();
```

3、用 function 来模拟参构造函数来实现（用 this 关键字定义构造的上下文属性）

```
function Pet(name,age,hobby){
    this.name=name;//this 作用域：当前对象
    this.age=age;
    this.hobby=hobby;
    this.eat=function(){
        alert("我叫"+this.name+",我喜欢"+this.hobby+",是个程序员");
    }
}
var maidou =new Pet("麦兜",25,"coding");//实例化、创建对象
```

```
maidou.eat();//调用 eat 方法
```

4、用工厂方式来创建（内置对象）

```
var wcDog =new Object();
wcDog.name="旺财";
wcDog.age=3;
wcDog.work=function(){
    alert("我是"+wcDog.name+",汪汪汪.....");
}
wcDog.work();
```

5、用原型方式来创建

```
function Dog(){
 }
Dog.prototype.name="旺财";
Dog.prototype.eat=function(){
 alert(this.name+"是个吃货");
}
var wangcai =new Dog();
wangcai.eat();
```

6、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();

10. 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；

11. 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 +')');

12. 什么是 window 对象？什么是 document 对象？

window 对象是指浏览器打开的窗口。

document 对象是 Documentd 对象（HTML 文档对象）的一个只读引用，window 对象的一个属性。

13. null，undefined 的区别？

null　　表示一个对象是“没有值”的值，也就是值为“空”；

undefined　　表示一个变量声明了没有初始化(赋值)；

undefined 不是一个有效的 JSON，而 null 是；

undefined 的类型(typeof)是 undefined；

null 的类型(typeof)是 object；

Javascript 将未赋值的变量默认值设为 undefined；

Javascript 从来不会将变量设为 null。它是用来让程序员表明某个用 var 声明的变量时没有值的。

typeof undefined

//"undefined"

undefined :是一个表示"无"的原始值或者说表示"缺少值"，就是此处应该有一个值，但是还没有定义。当尝试读取时会返回 undefined；

例如变量被声明了，但没有赋值时，就等于 undefined

typeof null

//"object"

null：是一个对象(空对象，没有任何属性和方法)；

例如作为函数的参数，表示该函数的参数不是对象；

注意：

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

null == undefined // true

null === undefined // false

再来一个例子：

null

Q：有张三这个人么？

A：有！

Q：张三有房子么？

A：没有！

undefined

Q：有张三这个人么？

A：有！

Q: 张三有多少岁？

A: 不知道（没有被告诉）

14. 写一个通用的事件监听器函数。

// event(事件)工具集，来源：github.com/markyun

markyun.Event = {

// 页面加载完成后

```
readyEvent : function(fn) {
    if (fn==null) {
        fn=document;
    }
    var oldonload = window.onload;
    if (typeof window.onload != 'function') {
        window.onload = fn;
    } else {
        window.onload = function() {
            oldonload();
            fn();
        };
    }
},
// 视能力分别使用 dom0||dom2||IE 方式 来绑定事件
// 参数： 操作的元素,事件名称 ,事件处理程序
addEvent : function(element, type, handler) {
    if (element.addEventListener) {
        //事件类型、需要执行的函数、是否捕捉
        element.addEventListener(type, handler, false);
    } else if (element.attachEvent) {
        element.attachEvent('on' + type, function() {
            handler.call(element);
        });
    } else {
        element['on' + type] = handler;
    }
```

```
},
// 移除事件
removeEvent : function(element, type, handler) {
    if (element.removeEventListener) {
        element.removeEventListener(type, handler, false);
    } else if (element.datachEvent) {
        element.detachEvent('on' + type, handler);
    } else {
        element['on' + type] = null;
    }
},
// 阻止事件 (主要是事件冒泡，因为 IE 不支持事件捕获)
stopPropagation : function(ev) {
    if (ev.stopPropagation) {
        ev.stopPropagation();
    } else {
        ev.cancelBubble = true;
    }
},
// 取消事件的默认行为
preventDefault : function(event) {
    if (event.preventDefault) {
        event.preventDefault();
    } else {
        event.returnValue = false;
    }
},
```

```
    // 获取事件目标
    getTarget : function(event) {
        return event.target || event.srcElement;
    },
    // 获取 event 对象的引用，取到事件的所有信息，确保随时能使用 event；
    getEvent : function(e) {
        var ev = e || window.event;
        if (!ev) {
            var c = this.getEvent.caller;
            while (c) {
                ev = c.arguments[0];
                if (ev && Event == ev.constructor) {
                    break;
                }
                c = c.caller;
            }
        }
        return ev;
    }
};
```

15. 事件是？IE 与火狐的事件机制有什么区别？ 如何阻止冒泡？

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

2. 事件处理机制：IE 是事件冒泡、Firefox 同时支持两种事件模型，也就是：捕获型事件和冒泡型事件；

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

16. 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');
    }
```

17. new 操作符具体干了什么呢?

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

2、属性和方法被加入到 this 引用的对象中。

3、新创建的对象由 this 所引用，并且最后隐式的返回 this 。

```
var obj  = {};
obj.__proto__ = Base.prototype;
Base.call(obj);
```

18. Javascript 中，有一个函数，执行时对象查找时，永远不会去查找原型，这个函数是?

hasOwnProperty

javaScript 中 hasOwnProperty 函数方法是返回一个布尔值，指出一个对象是否具有指定名称的属性。此方法无法检查该对象的原型链中是否具有该属性；该属性必须是对象本身的一个成员。

使用方法：

object.hasOwnProperty(proName)

其中参数 object 是必选项。一个对象的实例。

proName 是必选项。一个属性名称的字符串值。

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

19. JSON 的了解?

JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。

它是基于 JavaScript 的一个子集。数据格式简单, 易于读写, 占用带宽小

如：{"age":"12", "name":"back"}

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);
```

20. 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
        };
    })();
```

21. 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

22. DOM 操作——怎样添加、移除、移动、复制、创建和查找节点？

（1）创建新节点

```
createDocumentFragment()    //创建一个 DOM 片段
```

createElement()　//创建一个具体的元素

createTextNode()　//创建一个文本节点

（2）添加、移除、替换、插入

appendChild()

removeChild()

replaceChild()

insertBefore() //在已有的子节点前插入一个新的子节点

（3）查找

getElementsByTagName()　//通过标签名称

getElementsByName()　//通过元素的 Name 属性的值(IE 容错能力较强，会得到一个数组，其中包括 id 等于 name 值的)

getElementById()　//通过元素 Id，唯一性

23..call() 和 .apply() 的区别？

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

注意：js 中的函数其实是对象，函数名是对 Function 对象的引用。

```
function add(a,b)
{
    alert(a+b);
}

function sub(a,b)
{
    alert(a-b);
}

add.call(sub,3,1);
```

24. 那些操作会造成内存泄漏？（虎牙）

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）

25. 用 js 实现千位分隔符?

```
function commafy(num) {
    return num && num
        .toString()
        .replace(/(\d)(?=(\d{3})+\.)/g, function($0, $1) {
            return $1 + ",";
        });
}
console.log(commafy(1234567.90)); //1,234,567.90
```

26. 检测浏览器版本版本有哪些方式？

功能检测、userAgent 特征检测

比如：navigator.userAgent

//"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_2) AppleWebKit/537.36

(KHTML, like Gecko) Chrome/41.0.2272.101 Safari/537.36"

27. 列举 IE 与其他浏览器不一样的特性？

事件不同之处：

触发事件的元素被认为是目标（target）。而在 IE 中，目标包含在 event 对象的 srcElement 属性；

获取字符代码、如果按键代表一个字符（shift、ctrl、alt 除外），IE 的 keyCode 会返回字符代码（Unicode），

DOM 中按键的代码和字符是分离的，要获取字符代码，需要使用 charCode 属性；

阻止某个事件的默认行为，IE 中阻止某个事件的默认行为，必须将 returnValue 属性设置为 false，Mozilla 中，

需要调用 preventDefault() 方法；

停止事件冒泡，IE 中阻止事件进一步冒泡，需要设置 cancelBubble 为 true，Mozzilla 中，

需要调用 stopPropagation()；

28. 什么叫优雅降级和渐进增强？

优雅降级：Web 站点在所有新式浏览器中都能正常工作，如果用户使用的是老式浏览器，

则代码会针对旧版本的 IE 进行降级处理了,使之在旧式浏览器上以某种形式降级体验却不至于完全不能用。

如：border-shadow

渐进增强：从被所有浏览器支持的基本功能开始，逐步地添加那些只有新版本浏览器才支持的功能,

向页面增加不影响基础浏览器的额外样式和功能的。当浏览器支持时，它们会自动地呈现出来并发挥作用。

如：默认使用 flash 上传，但如果浏览器支持 HTML5 的文件上传功能，则使用 HTML5 实现更好的体验；

29. 请写出一下代码的运行结果

```
var f= true

if(f===true){

    var a = 10

}


function fn(){

    var b = 20
```

```
    c = 30
}

fn()
console.log(a)  //10
console.log(b)  //报错 b is not defined ,所以不会输出 b,c，如果再函数前面 var b，则会出现 undefined
console.log(c)
```

30. 输出今天的日期，以 YYYY-MM-DD 的方式，比如今天是 2015 年 7 月 7 日，则输出 2015-07-07

```
var d = new Date();
    // 获取年，getFullYear()返回 4 位的数字
var year = d.getFullYear();
    // 获取月，月份比较特殊，0 是 1 月，11 是 12 月
var month = d.getMonth() + 1;
    // 变成两位
month = month < 10 ? '0' + month : month;
    // 获取日
var day = d.getDate();
day = day < 10 ? '0' + day : day;

console.log(year + '-' + month + '-' + day)
```

31. 看下列代码，将会输出什么(考察点：声明的变量提升，IIFE)

```
var foo = 1  ;
(function(){
    console.log(foo)  //undefined
    var foo = 2
    console.log(foo)  //2
})()
```

32. 如何添加 HTML 事件，有几种方法？（至少两种方式）

1）通过 HTML 元素属性

name

2)通过对象属性

对象指的是 DOM 树里的对象，我们都知道，所有的 html 元素在 DOM(文档对象类型)里都存在一个相应的 DOM 元素。

html 结构:

name

js：

document.getElementById('n').onclick = function(){语句}

3)通过 W3C 监听方式（标准方式）或者 IE 专属的中间模型添加事件

W3C 方式：elemennt.addEventListener(事件名,处理函数引用,true || false)

true 表示捕获事件，false 冒泡阶段捕获事件，一般我们设为 false

IE 模式: element.attachEvent('on' + 事件名，处理函数引用)

兼容的模型：

```
function bind(obj,eventName,fn){
    if(obj.addEventListener){
        //标准浏览器
        obj.addEventListener(eventName,fn，false)
    }else{
        //非标准浏览器(ie 678)
        obj.attachEvent('on'+eventName,function(){
            fn.call(obj)
        })
    }
}
```

33. javascript 的 typeof 返回哪些数据类型

```
alert(typeof [1, 2]); //object

alert(typeof 'leipeng'); //string

var i = true;

alert(typeof i); //boolean

alert(typeof 1); //number

var a;

alert(typeof a); //undefined

function a(){;};

alert(typeof a) //function
```

34. 例举 3 种强制类型转换和 2 种隐式类型转换?

强制（parseInt(),parseFloat(),Number()）

隐式（== ,!!）

35. split() 、join() 的区别

前者是切割成数组的形式，后者是将数组转换成字符串

36. 数组方法 pop() push() unshift() shift()

push()尾部添加 pop()尾部删除

unshift()头部添加 shift()头部删除

map()：遍历数组中的元素，返回一个新数组(包含回调函数返回的数据)

filter():遍历数组中的元素，返回一个新数组(包含回调函数返回 true 的元素)

37. 事件绑定和普通事件有什么区别

普通添加事件的方法：

```
var btn = document.getElementById("hello");

btn.onclick = function(){

    alert(1);

}

btn.onclick = function(){

    alert(2);

}
```

执行上面的代码只会 alert 2

事件绑定方式添加事件：

```
var btn = document.getElementById("hello");
btn.addEventListener("click",function(){
    alert(1);
},false);
btn.addEventListener("click",function(){
    alert(2);
},false);
```

执行上面的代码会先 alert 1 再 alert 2

普通添加事件的方法不支持添加多个事件，最下面的事件会覆盖上面的，而事件绑定（addEventListener）方式添加事件可以添加多个。

addEventListener 不兼容低版本 IE

普通事件无法取消

addEventLisntener 还支持事件冒泡+事件捕获

38. IE 和 DOM 事件流的区别

1.执行顺序不一样、

2.参数不一样

3.事件加不加 on

4.this 指向问题

39. IE 和标准下有哪些兼容性的写法

```
var ev = ev || window.event
document.documentElement.clientWidth || document.body.clientWidth
var target = ev.srcElement||ev.target
```

call 和 apply 的区别

功能一样，都是将当前函数作为指定对象的方法执行，即函数中的 this 是指定对象

call(thisObj，arg1,arg2...)　//将所有参数一个一个传递进去

apply(thisObj，[argArray])　　//将所有参数放在数组中传入

40. Worker 继承 Person 的方法

```
//使用构造函数+原型的组合模式
function Person( age, name ){
    this.age = age;
    this.name = name;
}
Person.prototype.show = function(){
    alert('父级方法');
}

function Worker(age,name,job){
    Person.apply( this, arguments );
    this.job = job;
}
Worker.prototype = new Person();

var Person = new Person(14,'张三 ');
var Worker = new Worker (25,'李四','程序员');
```

41. 如何阻止事件冒泡和事件默认行为

```
//阻止事件冒泡
if(typeof ev.stopPropagation=='function') {　//标准的
    ev.stopPropagation();
} else { //非标准 IE
    window.event.cancelBubble = true;
}
```

//阻止事件默认行为

return false

42. 添加 删除 替换 插入到某个元素的方法

element.appendChild()

element.insertBefore()

element.replaceChild()

element.removeChild()

43. javascript 的内置对象和宿主对象

内置对象为 Object, Array, Function, Date, Math 等

宿主为浏览器自带的 window 等

44. window.onload 和 document ready 的区别

window.onload 是在 dom 文档树加载完和所有文件加载完之后执行一个函数 document.ready 原生中没有这个方法，jquery 中有 $().ready(function),在 dom 文档树加载完之后执行一个函数（注意，这里面的文档树加载完不代表全部文件加载完）。

$(document).ready 要比 window.onload 先执行

window.onload 只能出来一次，$(document).ready 可以出现多次

45. ”==”和“===”的不同

前者会自动转换类型

后者不会

46. 浏览器的同源策略

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

47. JavaScript 是一门什么样的语言，它有哪些特点？

javaScript 一种直译式脚本语言，是一种动态类型、弱类型、基于原型的语言，内置支持类型。它的解释器被称为 JavaScript 引擎，为浏览器的一部分，广泛用于客户端的脚本语言，最早是在 HTML 网页上使用，用来给 HTML 网页增加动态功能。JavaScript 兼容于 ECMA 标准，因此也称为 ECMAScript。

基本特点

1. 是一种解释性脚本语言（代码不进行预编译）。
2. 主要用来向 HTML（标准通用标记语言下的一个应用）页面添加交互行为。
3. 可以直接嵌入 HTML 页面，但写成单独的 js 文件有利于结构和行为的分离。

跨平台特性，在绝大多数浏览器的支持下，可以在多种平台下运行（如 Windows、Linux、Mac、Android、iOS 等）。

48. JavaScript 的数据类型都有什么？

基本数据类型：String,boolean,Number,Undefined, Null

引用数据类型：Object, Array, Function

那么问题来了，如何判断某变量是否为数组数据类型？

方法一.判断其是否具有“数组性质”，如 slice()方法。可自己给该变量定义 slice 方法，故有时会失效

方法二.obj instanceof Array 在某些 IE 版本中不正确

方法三.方法一二皆有漏洞，在 ECMA Script5 中定义了新方法 Array.isArray()，保证其兼容性，最好的方法如下：

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

49. 已知 ID 的 Input 输入框，希望获取这个输入框的输入值，怎么做？（不使用第三方框架）

```
document.getElementById(ID).value
```

50. 希望获取到页面中所有的 checkbox 怎么做？（不使用第三方框架）

```
var domList = document.getElementsByTagName(‘input’)
var checkBoxList = [];
var len = domList.length;      //缓存到局部变量
for (var i=0;i<len;i++) {
    if (domList[i].type ==  ‘checkbox’) {
        checkBoxList.push(domList[i]);
    }
}
```

51. 设置一个已知 ID 的 DIV 的 html 内容为 xxxx，字体颜色设置为黑色(不使用第三方框架)

var dom = document.getElementById(“ID”);

dom.innerHTML = “xxxx”

dom.style.color = “#000”

52. 当一个 DOM 节点被点击时候，我们希望能够执行一个函数，应该怎么做？

直接在 DOM 里绑定事件：

在 JS 里通过 onclick 绑定：xxx.onclick = test

通过事件添加进行绑定：addEventListener(xxx, ‘click’, test)

那么问题来了，Javascript 的事件流模型都有什么？

“事件冒泡”：事件开始由最具体的元素接受，然后逐级向上传播

“事件捕捉”：事件由最不具体的节点先接收，然后逐级向下，一直到最具体的

“DOM 事件流”：三个阶段：事件捕捉，目标阶段，事件冒泡

53. 看下列代码输出为何？解释原因。

var a;

alert(typeof a); // undefined

alert(b); // 报错

解释：Undefined 是一个只有一个值的数据类型，这个值就是“undefined”，在使用 var 声明变量但并未对其赋值进行初始化时，这个变量的值就是 undefined。而 b 由于未声明将报错。注意未申明的变量和声明了未赋值的是不一样的。

54. 看下列代码,输出什么？解释原因。

var a = null;

alert(typeof a); //object

解释：null 是一个只有一个值的数据类型，这个值就是 null。表示一个空指针对象，所以用 typeof 检测会返回”object”。

55. 看下列代码,输出什么？解释原因。

```
var undefined;

undefined == null; // true

1 == true;      // true

2 == true;      // false

0 == false;    // true

0 == '';         // true

NaN == NaN;    // false

[] == false; // true

[] == ![];      // true
```

undefined 与 null 相等，但不恒等（===）

一个是 number 一个是 string 时，会尝试将 string 转换为 number

尝试将 boolean 转换为 number，0 或 1

尝试将 Object 转换成 number 或 string，取决于另外一个对比量的类型

所以，对于 0、空字符串的判断，建议使用 “===” 。“===”会先判断两边的值类型，类型不匹配时为 false。

那么问题来了，看下面的代码，输出什么，foo 的值为什么？

```
var foo = "11"+2-"1";

console.log(foo);

console.log(typeof foo);
```

执行完后 foo 的值为 111，foo 的类型为 String。

56. 看代码给答案。

```
var a = new Object();

a.value = 1;

b = a;

b.value = 2;

alert(a.value);
```

答案：2（考察引用数据类型细节）

57.已知数组 var stringArray = [“This”, “is”, “Baidu”, “Campus”]，Alert 出”This is Baidu Campus”。

alert(stringArray.join(“”))

58.已知有字符串 foo=”get-element-by-id”,写一个 function 将其转化成驼峰表示法”getElementById”。

```
function combo(msg){

  var arr=msg.split("-");

  for(var i=1;i<arr.length;i++){

   arr[i]=arr[i].charAt(0).toUpperCase()+arr[i]

.substr(1,arr[i].length-1);

  }

  msg=arr.join("");

  return msg;

}
```

59.var numberArray = [3,6,2,4,1,5]; （考察基础 API）

1) 实现对该数组的倒排，输出[5,1,4,2,6,3]

numberArray.reverse()

2) 实现对该数组的降序排列，输出[6,5,4,3,2,1]

numberArray.sort(function(a,b){return b-a})

60.输出今天的日期，以 YYYY-MM-DD 的方式，比如今天是 2014 年 9 月 26 日，则输出 2014-09-26

```
var d = new Date();

// 获取年，getFullYear()返回 4 位的数字

var year = d.getFullYear();

// 获取月，月份比较特殊，0 是 1 月，11 是 12 月

var month = d.getMonth() + 1;

// 变成两位

month = month < 10 ? '0' + month : month;

// 获取日
```

```
var day = d.getDate();

day = day < 10 ? '0' + day : day;

alert(year + '-' + month + '-' + day);
```

61. 看下列代码，将会输出什么?(变量声明提升)

```
var foo = 1;

(function(){

    console.log(foo);

    var foo = 2;

    console.log(foo);

})()
```

答案：输出 undefined 和 2。上面代码相当于：

```
var foo = 1;

(function(){

    var foo;

    console.log(foo); //undefined

    foo = 2;

    console.log(foo); // 2;

})()
```

函数声明与变量声明会被 JavaScript 引擎隐式地提升到当前作用域的顶部，但是只提升名称不会提升赋值部分。

62. 把两个数组合并，并删除第二个元素。

```
var array1 = ['a','b','c'];

var bArray = ['d','e','f'];

var cArray = array1.concat(bArray);        cArray.splice(1,1);
```

63. 怎样添加、移除、移动、复制、创建和查找节点（原生 JS，实在基础，没细写每一步）

1）创建新节点

```
createDocumentFragment()  //创建一个 DOM 片段

createElement()  //创建一个具体的元素
```

createTextNode() //创建一个文本节点

2）添加、移除、替换、插入

appendChild() //添加

removeChild() //移除

replaceChild() //替换

insertBefore() //插入

3）查找

getElementsByTagName() //通过标签名称

getElementsByName() //通过元素的 Name 属性的值

getElementById() //通过元素 Id，唯一性

64.正则表达式构造函数 var reg=new RegExp(“xxx”)与正则表达字面量 var reg=//有什么不同？匹配邮箱的正则表达式？

当使用 RegExp()构造函数的时候，不仅需要转义引号（即\”表示”），并且还需要双反斜杠（即\\表示一个\）。使用正则表达字面量的效率更高。

邮箱的正则匹配：

```
var regMail = /^([a-zA-Z0-9_-])+@([a-zA-Z0-9_-])+((.[a-zA-Z0-9_-]{2,3}){1,2})$/;
```

65.看下面代码，给出输出结果。

```
for(var i=1;i<=3;i++){
  setTimeout(function(){
     console.log(i);
  },0);
};
```

答案：4 4 4。

原因：回调函数是在 for 结束之后才运行的。追问，如何让上述代码输出 1 2 3？

```
for(var i=1;i<=3;i++){
   setTimeout((function(j){  //改成立即执行函数
console.log(j);
   })(i),0);
};
```

66.写一个 function，清除字符串前后的空格。（兼容所有浏览器）

```
if (!String.prototype.trim) {
    String.prototype.trim = function() {
    return this.replace(/^\s+/, "").replace(/\s+$/,"");
}
}
//测试
var str = " \t\n test string ".trim();
alert(str == "test string"); // alerts "true"
```

67.Javascript 中，以下哪条语句一定会产生运行错误？

var _变量=NaN;　B、var 0bj = [];　C、var obj = //;　D、var obj = {};　答案(B C)

68.以下两个变量 a 和 b，a+b 的哪个结果是 NaN？　答案(　AC　)

A、var a=undefined; b=NaN

B、var a= ‘123’; b=NaN

C、var a =undefined , b =NaN

var a=NaN , b='undefined'

69.var a=10; b=20; c=4; ++b+c+a++ 以下哪个结果是正确的？

A 34　B、35　C、36　D、37　答案(　B　)

70.下面的 JavaScript 语句中，（ D ）实现检索当前页面中的表单元素中的所有文本框，并将它们全部清空

```
A. for(vari=0;i< form1.elements.length;i++) {
    if(form1.elements.type==”text”)
        form1.elements.value=”";}
B. for(vari=0;i<document.forms.length;i++) {
    if(forms[0].elements.type==”text”)
        forms[0].elements.value=”";
    }
```

C. if(document.form.elements.type==”text”)

form.elements.value=”";

D. for(vari=0;i<document.forms.length; i++){

for(var j=0;j<document.forms.elements.length; j++){

if(document.forms.elements[j].type==”text”)

document.forms.elements[j].value=”";

}

}

71.要将页面的状态栏中显示“已经选中该文本框”，下列 JavaScript 语句正确的是（ A ）

A. window.status=”已经选中该文本框”

B. document.status=”已经选中该文本框”

C. window.screen=”已经选中该文本框”

D. document.screen=”已经选中该文本框”

72.以下哪条语句会产生运行错误：（AD）

A.var obj = ();B.var obj = [];C.var obj = {};D.var obj = //;

73.以下哪个单词不属于 javascript 保留字：（B）

A.withB.parentC.classD.void

74.请选择结果为真的表达式：（C）

A.null instanceof ObjectB.null === undefinedC.null == undefinedD.NaN == NaN

75.typeof 运算符返回值中有一个跟 javascript 数据类型不一致,它是 Array。

76.定义了一个变量，但没有为该变量赋值，如果 alert 该变量，javascript 弹出的对话框中显示 undefined 。

77.分析代码，得出正确的结果。

var a=10, b=20 , c=30;

++a;

a++;

```
e=++a + (++b) + (c++) + a++;

alert(e);
```

弹出提示对话框：77

78. 写出函数 DateDemo 的返回结果，系统时间假定为今天

```
function DateDemo(){

  var d, s="今天日期是：";

  d = new Date();

s += d.getMonth() +1+ "/";

s += d.getDate() + "/";

s += d.getFullYear();

return s;

}
```

结果：今天日期是：7/21/2016

79. 写出程序运行的结果？

```
for(i=0, j=0; i<10, j<6; i++, j++){

k = i + j;}
```

结果：10

80. 阅读以下代码，请分析出结果：

```
var arr = new Array(1 ,3 ,5);

arr[4]='z';

arr2 = arr.reverse();

arr3 = arr.concat(arr2);

alert(arr3);
```

弹出提示对话框：z,,5,3,1,z,,5,3,1

81. 写出简单描述 html 标签（不带属性的开始标签和结束标签）的正则表达式，并将以下字符串中的 html 标签去除掉

var str = “ 这里是 div 里面的段落 ”;

<scripttype=”text/javascript”>

```
var reg = /<\/?\w+\/?>/gi;

var str = “<div>这里是 div<p>里面的段落</p></div>”;

alert(str.replace(reg,”"));

</script>
```

82. 截取字符串 abcdefg 的 efg

```
alert('abcdefg'.substring(4));
```

83. 列举浏览器对象模型 BOM 里常用的至少 4 个对象，并列举 window 对象的常用方法至少 5 个

对象：window, document, location, screen, history, navigator

方法：alert(), confirm(), prompt(), open(), close()

84. 简述列举文档对象模型 DOM 里 document 的常用的查找访问节点的方法并做简单说明

document.getElementById 根据元素 id 查找元素

document.getElementByName 根据元素 name 查找元素

document.getElementTagName 根据指定的元素名查找元素

85. 简述创建函数的几种方式（腾讯）

第一种（函数声明）：

```
function sum1(num1,num2){
   return num1+num2;
}
```

第二种（函数表达式）：

```
var sum2 = function(num1,num2){
   return num1+num2;
}
```

第三种（函数对象方式）：

```
var sum3 = new Function("num1","num2","return num1+num2");
```

86. Javascript 如何实现继承？

1.构造继承法

2.原型继承法

3.实例继承法

87. Javascript 创建对象的几种方式？

1、var obj = {};（使用 json 创建对象）

如：obj.name = '张三';

obj.action = function ()

{

alert('吃饭');

};

2、var obj = new Object();（使用 Object 创建对象）

如：obj.name = '张三';

obj.action = function ()

{

alert('吃饭');

};

3、通过构造函数创建对象。

(1)、使用 this 关键字

如：var obj = function (){

this.name ='张三';

this.age = 19;

this.action = function ()

{

alert('吃饭');

};

}

(2)、使用 prototype 关键字

如：function obj (){}

```
    obj.prototype.name ='张三';
obj.prototype.action=function ()
{
     alert('吃饭');
};
```

4、使用内置对象创建对象。

如：var str = new String("实例初始化 String");

var str1 = "直接赋值的 String";

var func = new Function("x","alert(x)");//示例初始化 func

var obj = new Object();//示例初始化一个 Object

88. iframe 的优缺点？

优点：

1. 解决加载缓慢的第三方内容如图标和广告等的加载问题

2. Security sandbox

3. 并行加载脚本

缺点：

1. iframe 会阻塞主页面的 Onload 事件

2. 即时内容为空，加载也需要时间

3. 没有语意

89. 请你谈谈 Cookie 的弊端？

1.Cookie 数量和长度的限制。每个 domain 最多只能有 20 条 cookie，每个 cookie 长度不能超过 4KB，否则会被截掉。

2.安全性问题。如果 cookie 被人拦截了，那人就可以取得所有的 session 信息。即使加密也与事无补，因为拦截者并不需要知道 cookie 的意义，他只要原样转发 cookie 就可以达到目的了。

3.有些状态不可能保存在客户端。例如，为了防止重复提交表单，我们需要在服务器端保存一个计数器。如果我们把这个计数器保存在客户端，那么它起不到任何作用。

90. js 延迟加载的方式有哪些？

1. defer 和 async

2. 动态创建 DOM 方式（创建 script，插入到 DOM 中，加载完毕后 callBack）

3. 按需异步载入 js

91. 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

92. 哪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。

1. 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。

2. 闭包

3. 控制台日志

4. 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）

93. 判断一个字符串中出现次数最多的字符，统计这个次数

```
var str = 'asdfssaaasasasasaa';

var json = {};

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

        if(!json[str.charAt(i)]){

                json[str.charAt(i)] = 1;

        }else{

                json[str.charAt(i)]++;

        }

};

var iMax = 0;

var iIndex = '';

for(var i in json){

        if(json[i]>iMax){

                iMax = json[i];

                iIndex = i;
```

```
            }
        }
        alert('出现次数最多的是:'+iIndex+'出现'+iMax+'次');
```

94. 写一个获取非行间样式的函数

```
function getStyle(obj,attr,value)
{
    if(!value)
    {
        if(obj.currentStyle)
        {
            return obj.currentStyle(attr);
        }
        else{
            obj.getComputedStyle(attr,false);
        }
    }
    else
    {
        obj.style[attr] = value;
    }
}
```

95. 事件委托是什么

利用事件冒泡的原理，让自己的所触发的事件，让他的父元素代替执行！

96. 闭包是什么，有什么特性，对页面有什么影响

当内部函数使用了外部函数的局部变量时，产生的一个对象(包含了所有使用了的变量)

作用：在函数执行完后，局部变量还会存在

```
function outer(){
```

```
        var num = 1;
        function inner(){
            var n = 2;
            alert(n + num);
        }
        return inner;
    }
    var r = outer();
    r();
```

97. 解释 jsonp 的原理，以及为什么不是真正的 ajax

动态创建 script 标签，回调函数

Ajax 是页面无刷新请求数据操作

98. 字符串反转，如将 ’12345678’ 变成 ’87654321’

//思路：先将字符串转换为数组 split()，利用数组的反序函数 reverse()颠倒数组，再利用 jion() 转换为字符串

```
var str = '12345678';
str = str.split('').reverse().join('');
```

99. 将数字 12345678 转化成 RMB 形式 如： 12,345,678

//思路：先将数字转为字符， str= str + '' ;

//利用反转函数，每三位字符加一个 ','最后一位不加； re()是自定义的反转函数，最后再反转回去！

```
function re(str) {
    str += '';
    return str.split("").reverse().join("");
}

function toRMB(num) {
    var tmp='';
    for (var  i  =  1;  i  <=  re(num).length;  i++) {
```

```
            tmp += re(num)[i - 1];
            if (i % 3 == 0 && i != re(num).length) {
                tmp += ',';
            }
        }
        return re(tmp);
    }
```

100. 生成 5 个不同的随机数

//思路：5 个不同的数，每生成一次就和前面的所有数字相比较，如果有相同的，则放弃当前生成的数字！

```
var num1 = [];
for(var i = 0; i < 5; i++){
  num1[i] = Math.floor(Math.random()*10) + 1; //范围是 [1, 10]
  for(var j = 0; j < i; j++){
    if(num1[i] == num1[j]){
      i--;
    }
  }
}
```

101. 去掉数组中重复的数字

方法一：

//思路：每遍历一次就和之前的所有做比较，不相等则放入新的数组中！

//这里用的原型 个人做法；

```
Array.prototype.unique = function(){
  var len = this.length,
    newArr = [],
    flag = 1;
  for(var i = 0; i < len; i++, flag = 1){
```

```
        for(var j = 0; j < i; j++){
            if(this[i] == this[j]){
                flag = 0;      //找到相同的数字后，不执行添加数据
            }
        }
        flag ? newArr.push(this[i]) : '';
    }
    return newArr;
}
```

方法二：

```
var arr=[1,2,3,3,4,4,5,5,6,1,9,3,25,4];
Array.prototype.unique2 = function()
{
    var n = []; //一个新的临时数组
    for(var i = 0; i < this.length; i++) //遍历当前数组
    {
        //如果当前数组的第 i 已经保存进了临时数组，那么跳过，
        //否则把当前项 push 到临时数组里面
        if (n.indexOf(this[i]) == -1) n.push(this[i]);
    }
    return n;
}
var newArr2=arr.unique2(arr);
alert(newArr2); //输出 1,2,3,4,5,6,9,25
```

102.阶乘函数

//原型方法

```
Number.prototype.N = function(){
   var re = 1;
   for(var i = 1; i <= this; i++){
      re *= i;
   }
   return re;
}
var num = 5;
alert(num.N());
```

103.window.location.reload() 作用？

刷新当前页面。

104.下面输出多少？

```
var o1 = new Object();
var o2 = o1;
o2.name = "CSSer";
console.log(o1.name);
```

如果不看答案，你回答真确了的话，那么说明你对 javascript 的数据类型了解的还是比较清楚了。js 中有两种数据类型，分别是：基本数据类型和引用数据类型（object Array）。对于保存基本类型值的变量，变量是按值访问的，因为我们操作的是变量实际保存的值。对于保存引用类型值的变量，变量是按引用访问的，我们操作的是变量值所引用（指向）的对象。答案就清楚了： //CSSer;

105.a 输出多少？

```
var a = 6;
```

```
setTimeout(function () {
    var a = 666;
    alert(a);    // 输出 666，
}, 1000);
```

因为 var a = 666;定义了局部变量 a，并且赋值为 666，根据变量作用域链，全局变量处在作用域末端，优先访问了局部变量，从而覆盖了全局变量 。

```
var a = 6;
setTimeout(function () {
    alert(a);    // 输出 undefined
    var a = 666;
}, 1000);
```

因为 var a = 666;定义了局部变量 a，同样覆盖了全局变量，但是在 alert(a);之前,a 并未赋值，所以输出 undefined。

```
var a = 6;
setTimeout(function(){
    alert(a);
    var a = 66;
}, 1000);
a = 666;
alert(a);
// 666, undefined;
```

记住： 异步处理，一切 OK 声明提前

106. 看程序，写结果

```
function setN(obj){
```

```
        obj.name='屌丝';
        obj = new Object();
        obj.name = '腐女';
    };
    var per = new Object();
    setN(per);
    alert(per.name);  //屌丝 内部
```

107. 加减运算

```
alert('5'+3); //53 string
alert('5'+'3'); //53 string
alert('5'-3); //2 number
alert('5'-'3'); //2 number
```

108. 为什么不能定义 1px 左右的 div 容器？

IE6 下这个问题是因为默认的行高造成的，解决的方法也有很多，例如：

overflow:hidden | zoom:0.08 | line-height:1px

109. 结果是什么？

```
function foo(){
   foo.a = function(){alert(1)};
   this.a = function(){alert(2)};
   a = function(){alert(3)};
   var a = function(){alert(4)};
};
foo.prototype.a = function(){alert(5)};
foo.a = function(){alert(6)};
foo.a(); //6
var obj = new foo();
obj.a(); //2
```

```
foo.a(); //1
```

110. 输出结果

```
var a = 5;

function test(){

   a = 0;

   alert(a);

   alert(this.a); //没有定义 a 这个属性

   var a;

   alert(a)

}

test(); // 0, 5, 0

new test(); // 0, undefined, 0 //由于类它自身没有属性 a， 所以是 undefined
```

111. 结果是：

```
    var bool = !!2; alert(bool)；//true;
```

双向非操作可以把字符串和数字转换为布尔值。

112. 声明对象，添加属性，输出属性

```
    var obj = {

        name: 'leipeng',

        showName: function(){

           alert(this.name);

        }

      }

    obj.showName();
```

113. 匹配输入的字符：第一个必须是字母或下划线开头，长度 5-20

```
var reg = /^[a-zA-Z_][a-zA-Z0-9_]{5,20}/,
```

```
        name1 = 'leipeng',
        name2 = '0leipeng',
        name3 = '你好 leipeng',
        name4 = 'hi';

    alert(reg.test(name1));
    alert(reg.test(name2));
    alert(reg.test(name3));
    alert(reg.test(name4));
```

114.检测变量类型

```
function checkStr(str){
    return str =='string';
}
console.log(checkStr("aaa"));
```

115.如何在 HTML 中添加事件，几种方法？

1、标签之中直接添加 onclick="fun()";

2、JS 添加 Eobj.onclick = method;

3、绑定事件 IE： obj.attachEvent('onclick', method)；

FF: obj.addEventListener('click', method, false);

116.BOM 对象有哪些，列举 window 对象？

1、window 对象 ，是 JS 的最顶层对象，其他的 BOM 对象都是 window 对象的属性；

2、document 对象，文档对象；

3、location 对象，浏览器当前 URL 信息；

4、navigator 对象，浏览器本身信息；

5、screen 对象，客户端屏幕信息；

6、history 对象，浏览器访问历史信息；

117.JS 中的简单继承 call 方法

```
//定义个父母类，注意：类名都是首字母大写的哦！
```

```
function Parent(name, money){
    this.name = name;
    this.money = money;
    this.info = function(){
        alert('姓名： '+this.name+' 钱： '+ this.money);
    }
}
//定义孩子类
function Children(name){
    Parent.call(this, name); //继承 姓名属性，不要钱。
    this.info = function(){
        alert('姓名： '+this.name);
    }
}
//实例化类
var per = new Parent('parent', 800000000000);
var chi = new Children('child');
per.info();
chi.info();
```

118.bind(), live(), delegate()的区别

bind： 绑定事件，对新添加的事件不起作用，方法用于将一个处理程序附加到每个匹配元素的事件上并返回 jQuery 对象。

live： 方法将一个事件处理程序附加到与当前选择器匹配的所有元素（包含现有的或将来添加的）的指定事件上并返回 jQuery 对象。

delegate：方法基于一组特定的根元素将处理程序附加到匹配选择器的所有元素（现有的或将来的）的一个或多个事件上。

119.看下列代码输出什么？

```
var foo = "11"+2-"1";

console.log(foo);

console.log(typeof foo);
```

执行完后 foo 的值为 111，foo 的类型为 Number。

120.看下列代码,输出什么？

```
var a = new Object();

a.value = 1;

b = a;

b.value = 2;

alert(a.value);
```

执行完后输出结果为 2

121.你如何优化自己的代码？

代码重用

避免全局变量（命名空间，封闭空间，模块化 mvc..）

拆分函数避免函数过于臃肿

注释

122.请描述出下列代码运行的结果

```
function d(){

console.log(this);

}

d();//输出 window 对象
```

123.以下代码中 end 字符串什么时候输出

```
var t=true;

setTimeout(function(){

    console.log(123);

    t=false;
```

```
},1000);

while(t){}

console.log(‘end’);
```

永远不输出

124. 请尽可能详尽的解释 ajax 的工作原理

Ajax 的工作原理相当于在用户和服务器之间加了一个中间层，使用户操作与服务器响应异步化。这样把以前的一些服务器负担的工作转嫁到客户端，利于客户端闲置的处理能力来处理，减轻服务器和带宽的负担，从而达到节约 ISP 的空间及带宽租用成本的目的。

简单来说通过 XmlHttpRequest 对象来向服务器发异步请求，从服务器获得数据，然后用 javascript 来操作 DOM 而更新页面。这其中最关键的一步就是从服务器获得请求数据。要清楚这个过程和原理，我们必须对 XMLHttpRequest 有所了解。

125. 什么是三元表达式？“三元”表示什么意思？

三元运算符:

三元如名字表示的三元运算符需要三个操作数。

语法是 条件 ？结果 1：结果 2;. 这里你把条件写在问号(?)的前面后面跟着用冒号(:)分隔的结果 1 和结果 2。满足条件时结果 1 否则结果 2。

126. 浏览器标准模式和怪异模式之间的区别是什么？

所谓的标准模式是指，浏览器按 W3C 标准解析执行代码；怪异模式则是使用浏览器自己的方式解析执行代码，因为不同浏览器解析执行的方式不一样，所以我们称之为怪异模式

127. 下面这段代码想要循环输出结果 01234，请问输出结果是否正确，如果不正确，请说明为什么，并修改循环内的代码使其输出正确结果

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

    setTimeout(function(){

            console.log(i+’’);

        },100*i);

}
```

128.关于 IE 的 window 对象表述正确的有：（ACD）

A. window.opener 属性本身就是指向 window 对象

B. window.reload()方法可以用来刷新当前页面　应该是 location.reload 或者 window.location.reload

C. window.location=”a.html”和 window.location.href=”a.html”的作用都是把当前页面替换成 a.html 页面

D. 定义了全局变量 g；可以用 window.g 的方式来存取该变量

129.下面正确的是　A

A: 跨域问题能通过 JsonP 方案解决

B：不同子域名间仅能通过修改 window.name 解决跨域　还可以通过 script 标签 src jsonp 等 h5 Java split 等

C：只有在 IE 中可通过 iframe 嵌套跨域

D：MediaQuery 属性是进行视频格式检测的属性是做响应式的

130.错误的是 B

A: Ajax 本质是 XMLHttpRequest

B: 块元素实际占用的宽度与它的 width、border、padding 属性有关，与 background 无关

C: position 属性 absolute、fixed、---relative---会使文档脱标

D: float 属性 left 也会使 div 脱标

答案 C：relative 不会脱离文档流

131.变量的命名规范以及命名推荐

变量，函数，方法：小写开头，以后的每个单词首字母大写 （驼峰）

构造函数，class：每个单词大写开头

基于实际情况，以动词，名词，谓词来命名。尽量言简意骇，以命名代替注释

132.三种弹窗的单词以及三种弹窗的功能

1.alert

```
//弹出对话框并输出一段提示信息
    function ale() {
        //弹出一个对话框
        alert("提示信息！");
```

```
    }
```

2.confirm

```
    //弹出一个询问框，有确定和取消按钮
    function firm() {
        //利用对话框返回的值 （true 或者 false）
        if (confirm("你确定提交吗？")) {
            alert("点击了确定");
        }
        else {
            alert("点击了取消");
        }

    }
```

3.prompt

```
    //弹出一个输入框，输入一段文字，可以提交
    function prom() {
        var name = prompt("请输入您的名字", ""); //将输入的内容赋给变量
name ，

        //这里需要注意的是，prompt 有两个参数，前面是提示的话，后面是当对
话框出来后，在对话框里的默认值
        if (name)//如果返回的有内容
        {
            alert("欢迎您：" + name)
        }
```

```
        }
```

133.console.log(8 | 1); 输出值是多少？

答案：9

134.jQuery 框架中$.ajax()的常用参数有哪些？写一个 post 请求并带有发送数据和返回数据的样例

async 是否异步

url 请求地址

contentType 发送信息至服务器时内容编码类型

data 发送到服务器的数据

dataType 预期服务器返回的数据类型

type 请求类型

success 请求成功回调函数

error 请求失败回调函数

```
$.ajax({
            url: "/jquery/test1.txt",
            type: 'post',
            data: {
                id: 1
            },
            success: function(data) {
                alert(data);
            }
        }
```

135.JavaScript 的循环语句有哪些？

for,for..in,while,do...while

136.闭包：下面这个 ul，如何点击每一列的时候 alert 其 index？

```
<ul id="test">
```

```
    <li>这是第一条</li>
    <li>这是第二条</li>
    <li>这是第三条</li>
</ul>
//js
  window.onload = function() {
        var lis = document.getElementById('test').children;
        for (var i = 0; i < lis.length; i++) {
            lis[i].onclick = (function(i) {
                return function() {
                    alert(i)
                };
            })(i);
        };
    }
```

137.用正则表达式，写出由字母开头，其余由数字、字母、下划线组成的 6~30 的字符串？

^[a-zA-Z]{1}[\w]{5,29}$

138.列举浏览器对象模型 BOM 里常用的至少 4 个对象，并列举 window 对象的常用方法至少 5 个

对象：Window document location screen history navigator

方法：Alert() confirm() prompt() open() close()

139.写一个函数可以计算 sum(5,0,-5);输出 0; sum(1,2,3,4);输出 10;

```
function sum() {
    var result = 0;
    var arr = arguments;
    for (var i = 0; i < arr.length; i++) {
        var num = arguments[i];
        if (typeof num=='number') {
```

```
                result += num;
            };
        };
        return result;
}
```

140.请写出一个程序，在页面加载完成后动态创建一个 form 表单，并在里面添加一个 input 对象并给它任意赋值后义 post 方式提交到：
http://127.0.0.1/save.php

```
window.onload=function(){
        var form=document.createElement("form");
        form.setAttribute("method", "post");
        form.setAttribute("action", "http://127.0.0.1/save.php");
        var input=document.createElement("input");
        form.appendChild(input);
        document.body.appendChild(form);
        input.value="cxc";
        form.submit();//提交表单
}
```

141.用 JavaScript 实现冒泡排序。数据为 23、45、18、37、92、13、24

```
//升序算法
```

```
function sort(arr){
    for (var i = 0; i <arr.length; i++) {
        for (var j = 0; j <arr.length-i; j++) {
            if(arr[j]>arr[j+1]){
                var c=arr[j];//交换两个变量的位置
                arr[j]=arr[j+1];
                arr[j+1]=c;
            }
        };
    };
    return arr.toString();
}
console.log(sort([23,45,18,37,92,13,24]));
```

142. 前端代码优化的方法

```
var User = {
    count = 1，
    getCount：function（）{
        return this.count;
    }
}
console.log(User.getCount());
var func = User.getCount;
console.log(func());
```

1 undefined（因为是 window 对象执行了 func 函数）；

143.下列 JavaScript 代码执行后，依次 alert 的结果是

```
(function test(){
        var a=b=5;
        alert(typeof a);
        alert(typeof b);
})();
alert(typeof a);
alert(typeof b);
```

答案：number

number

undefined

Number

145.下列 JavaScript 代码执行后，iNum 的值是

```
var iNum = 0;
for(var i = 1; i< 10; i++){
      if(i % 5 == 0){
            continue;
     }
     iNum++;
}
```

答案：8

146.下列 JavaScript 代码执行后，依次 alert 的结果是

```
var obj = {proto: {a:1,b:2}};
function F(){};
F.prototype = obj.proto;
var f = new F();
obj.proto.c = 3;
obj.proto = {a:-1, b:-2};
```

```
alert(f.a);

alert(f.c);

delete F.prototype['a'];

alert(f.a);

alert(obj.proto.a);
```

答案：

1

3

undefined

-1

147.下列 JavaScript 代码执行后的 li 元素的数量是

```
<ul>
    <li>Item</li>
    <li></li>
    <li></li>
    <li>Item</li>
    <li>Item</li>
</ul>


var items = document.getElementsByTagName('li');
for(var i = 0; i< items.length; i++){
    if(items[i].innerHTML == ''){
        items[i].parentNode.removeChild(items[i]);
    }
}
```

148.程序中捕获异常的方法？

window.error

try{}catch(){}finally{}

149.给 String 对象添加一个方法，传入一个 string 类型的参数，然后将 string 的每个字符间价格空格返回，例如：addSpace(“hello world”) // -> ‘h e l l o ?w o r l d’

```
String.prototype.spacify = function(){

return this.split('').join(' ');

};
```

150.下列控制台都输出什么

第 1 题：

```
function setName(){

    name="张三";

}

setName();

console.log(name);
```

答案："张三"

第 2 题：

```
//考点：1、变量声明提升  2、变量搜索机制

var a=1;

function test(){

    console.log(a);

    var a=1;

}

test();
```

答案：undefined

第 3 题：

```
var b=2;

function test2(){

    window.b=3;

    console.log(b);

}
```

```
test2();
```

答案：3

第 4 题：

```
c=5;//声明一个全局变量 c
function test3(){
    window.c=3;
    console.log(c);          //答案：undefined，原因：由于此时的 c 是一个局部变量 c，并且没有被赋值
    var c;
    console.log(window.c);//答案：3，原因：这里的 c 就是一个全局变量 c
}
test3();
```

第 5 题：

```
var arr = [];
arr[0]  = 'a';
arr[1]  = 'b';
arr[10] = 'c';
alert(arr.length);  //答案：11
console.log(arr[5]);    //答案：undefined
```

第 6 题：

```
var a=1;
console.log(a++);       //答案：1
console.log(++a);       //答案：3
```

第 7 题：

```
console.log(null==undefined);   //答案：true
console.log("1"==1);        //答案：true，因为会将数字 1 先转换为字符串 1
```

```
console.log("1"===1);        //答案：false，因为数据类型不一致
```

第 8 题：

```
typeof 1;       "number"
typeof "hello";             "string"
typeof /[0-9]/;             "object"
typeof {};      "object"
typeof null;            "object"
typeof undefined; "undefined"
typeof [1,2,3];             "object"
typeof function(){};      //"function"
```

第 9 题：

```
parseInt(3.14);                 //3
parseFloat("3asdf");            //3
parseInt("1.23abc456");
parseInt(true);//"true" NaN
```

第 10 题：

```
//考点：函数声明提前
function bar() {
    return foo;
    foo = 10;
    function foo() {}
    //var foo = 11;
```

```
}
alert(typeof bar());//"function"
```

第 11 题：

```
//考点：函数声明提前
var foo = 1;
function bar() {
    foo = 10;
    return;
    function foo() {}
}
bar();
alert(foo);//答案：1
```

第 12 题：

```
console.log(a);//是一个函数
var a = 3;
function a(){}
console.log(a);////3
```

第 13 题：

```
//考点：对 arguments 的操作
function foo(a) {
    arguments[0] = 2;
    alert(a);//答案：2，因为：a、arguments 是对实参的访问，b、通过 arguments[i]
可以修改指定实参的值
}
foo(1);
```

第 14 题：

```
function foo(a) {
    alert(arguments.length);//答案：3，因为 arguments 是对实参的访问
```

```
}
foo(1, 2, 3);
```

第 15 题

```
bar();//报错
var foo = function bar(name) {
    console.log("hello"+name);
    console.log(bar);
};
//alert(typeof bar);
foo("world");//"hello"
console.log(bar);//undefined
console.log(foo.toString());
bar();//报错
```

第 16 题：

```
function test(){
    console.log("test 函数");
}
setTimeout(function(){
    console.log("定时器回调函数");
}, 0)
test();
```

结果：

test 函数

定时器回调函数

四、AJAX

1.Ajax 是什么？如何创建一个 Ajax？

ajax 的全称：Asynchronous Javascript And XML。

异步传输+js+xml。

所谓异步，在这里简单地解释就是：向服务器发送请求的时候，我们不必等待结果，而是可以同时做其他的事情，等到有了结果它自己会根据设定进行后续操作，与此同时，页面是不会发生整页刷新的，提高了用户体验。

(1)创建 XMLHttpRequest 对象,也就是创建一个异步调用对象

(2)创建一个新的 HTTP 请求,并指定该 HTTP 请求的方法、URL 及验证信息

(3)设置响应 HTTP 请求状态变化的函数

(4)发送 HTTP 请求

(5)获取异步调用返回的数据

(6)使用 JavaScript 和 DOM 实现局部刷新

2.Ajax 解决浏览器缓存问题？

1、在 ajax 发送请求前加上 anyAjaxObj.setRequestHeader("If-Modified-Since","0")。

2、在 ajax 发送请求前加上 anyAjaxObj.setRequestHeader("Cache-Control","no-cache")。

3、在 URL 后面加上一个随机数： "fresh=" + Math.random();。

4、在 URL 后面加上时间搓："nowtime=" + new Date().getTime();。

5、如果是使用 jQuery，直接这样就可以了 $.ajaxSetup({cache:false})。

这样页面的所有 ajax 都会执行这条语句就是不需要保存缓存记录。

3. 同步和异步的区别?

同步的概念应该是来自于 OS 中关于同步的概念:不同进程为协同完成某项工作而在先后次序上调整(通过阻塞,唤醒等方式).

同步强调的是顺序性.谁先谁后.异步则不存在这种顺序性.

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

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

4. 如何解决跨域问题?

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

5.http 状态码有那些？分别代表是什么意思？

简单版

[

100 Continue 继续，一般在发送 post 请求时，已发送了 http header 之后服务端将返回此信息，表示确认，之后发送具体参数信息

200 OK 正常返回信息

201 Created 请求成功并且服务器创建了新的资源

202 Accepted 服务器已接受请求，但尚未处理

301 Moved Permanently 请求的网页已永久移动到新位置。

302 Found 临时性重定向。

303 See Other 临时性重定向，且总是使用 GET 请求新的 URI。

304 Not Modified 自从上次请求后，请求的网页未修改过。

400 Bad Request 服务器无法理解请求的格式，客户端不应当尝试再次使用相同的内容发起请求。

401 Unauthorized 请求未授权。

403 Forbidden 禁止访问。

404 Not Found 找不到如何与 URI 相匹配的资源。

500 Internal Server Error　最常见的服务器端错误。

503 Service Unavailable 服务器端暂时无法处理请求（可能是过载或维护）。

]

完整版

1**(信息类)：表示接收到请求并且继续处理

100——客户必须继续发出请求

101——客户要求服务器根据请求转换 HTTP 协议版本

2**(响应成功)：表示动作被成功接收、理解和接受

200——表明该请求被成功地完成，所请求的资源发送回客户端

201——提示知道新文件的 URL

202——接受和处理、但处理未完成

203——返回信息不确定或不完整

204——请求收到，但返回信息为空

205——服务器完成了请求，用户代理必须复位当前已经浏览过的文件

206——服务器已经完成了部分用户的 GET 请求

3**(重定向类)：为了完成指定的动作，必须接受进一步处理

300——请求的资源可在多处得到

301——本网页被永久性转移到另一个 URL

302——请求的网页被转移到一个新的地址，但客户访问仍继续通过原始 URL 地址，重定向，新的 URL 会在 response 中的 Location 中返回，浏览器将会使用新的 URL 发出新的 Request。

303——建议客户访问其他 URL 或访问方式

304——自从上次请求后，请求的网页未修改过，服务器返回此响应时，不会返回网页内容，代表上次的文档已经被缓存了，还可以继续使用

305——请求的资源必须从服务器指定的地址得到

306——前一版本 HTTP 中使用的代码，现行版本中不再使用

307——申明请求的资源临时性删除

4**(客户端错误类)：请求包含错误语法或不能正确执行

400——客户端请求有语法错误，不能被服务器所理解

401——请求未经授权，这个状态代码必须和 WWW-Authenticate 报头域一起使用

HTTP 401.1 - 未授权：登录失败

HTTP 401.2 - 未授权：服务器配置问题导致登录失败

HTTP 401.3 - ACL 禁止访问资源

HTTP 401.4 - 未授权：授权被筛选器拒绝

HTTP 401.5 - 未授权：ISAPI 或 CGI 授权失败

402——保留有效 ChargeTo 头响应

403——禁止访问，服务器收到请求，但是拒绝提供服务

HTTP 403.1 禁止访问：禁止可执行访问

HTTP 403.2 - 禁止访问：禁止读访问

HTTP 403.3 - 禁止访问：禁止写访问

HTTP 403.4 - 禁止访问：要求 SSL

HTTP 403.5 - 禁止访问：要求 SSL 128

HTTP 403.6 - 禁止访问：IP 地址被拒绝

HTTP 403.7 - 禁止访问：要求客户证书

HTTP 403.8 - 禁止访问：禁止站点访问

HTTP 403.9 - 禁止访问：连接的用户过多

HTTP 403.10 - 禁止访问：配置无效

HTTP 403.11 - 禁止访问：密码更改

HTTP 403.12 - 禁止访问：映射器拒绝访问

HTTP 403.13 - 禁止访问：客户证书已被吊销

HTTP 403.15 - 禁止访问：客户访问许可过多

HTTP 403.16 - 禁止访问：客户证书不可信或者无效

HTTP 403.17 - 禁止访问：客户证书已经到期或者尚未生效

404——一个 404 错误表明可连接服务器，但服务器无法取得所请求的网页，请求资源不存在。eg：输入了错误的 URL

405——用户在 Request-Line 字段定义的方法不允许

406——根据用户发送的 Accept 拖，请求资源不可访问

407——类似 401，用户必须首先在代理服务器上得到授权

408——客户端没有在用户指定的饿时间内完成请求

409——对当前资源状态，请求不能完成

410——服务器上不再有此资源且无进一步的参考地址

411——服务器拒绝用户定义的 Content-Length 属性请求

412——一个或多个请求头字段在当前请求中错误

413——请求的资源大于服务器允许的大小

414——请求的资源 URL 长于服务器允许的长度

415——请求资源不支持请求项目格式

416——请求中包含 Range 请求头字段，在当前请求资源范围内没有 range 指示值，请求也不包含 If-Range 请求头字段

417——服务器不满足请求 Expect 头字段指定的期望值，如果是代理服务器，可能是下一级服务器不能满足请求长。

5**(服务端错误类)：服务器不能正确执行一个正确的请求

HTTP 500 - 服务器遇到错误，无法完成请求

HTTP 500.100 - 内部服务器错误 - ASP 错误

HTTP 500-11 服务器关闭

HTTP 500-12 应用程序重新启动

HTTP 500-13 - 服务器太忙

HTTP 500-14 - 应用程序无效

HTTP 500-15 - 不允许请求 global.asa

Error 501 - 未实现

HTTP 502 - 网关错误

HTTP 503：由于超载或停机维护，服务器目前无法使用，一段时间后可能恢复正常

5.什么是 Ajax,它们的优缺点

Ajax（Asynchronous JavaScript and XML）是一种浏览器端不用刷新整个页面就可以与服务器端通信的技术

它不是新技术, 而是一种由多种技术组合的技术,包括 Javascript、HTML 和 CSS、DOM、XML 和 JSON、XMLHttpRequest

HTML，CSS 用于呈现

DOM 实现动态显示和交互

XML 和 JSON 进行数据交换与处理

XMLHttpRequest 对象用于进行异步请求数据读取

Javascript 绑定和处理所有数据

优点：

减轻服务器的负担，Ajax 一般只从服务器获取只需要的数据。

无需刷新整个页面, 减少用户等待时间。

更好的客户体验，可以将一些服务器的工作转移到客户端完成，节约网络资源，提高用户体验。

基于标准化的对象，不需要安装特定的插件, 浏览器都能支持 Ajax

彻底将页面与数据分离。

缺点：

没有浏览历史, 不能回退,可能造成请求数的增加

存在跨域请求问题

对搜索引擎支持比较弱

6. 跨域问题

同源策略是浏览器的一种安全策略，所谓同源是指，域名，协议，端口完全相同

1)JSONP（JSON with Padding）

只支持 get 请求，不支持 post 请求

编码:客户端 url?callback=? 和 服务器端:var callback = req.query.callback

返回的 value 需要封装成 json

res.send(callback+'('+json+')');

背后:使用<srcipt>发送请求 和 只能发 get 请求

<srcipt>发送请求实现过程：

动态生成<script>标签，然后通过标签的 src 属性发送 GET 类型的跨域请求，

服务器端返回一段 js 脚本，这段 js 脚本的内容为一个函数调用,实参为需要返回的响应数据(json),

客户端这边需要提前定义好对应的函数，当<script>请求成功并接收到数据时，

会自动调用此函数，在函数中 我们就可以处理响应数据了

2)CORS

Cross-Origin Resource Sharing(跨域资源共享)

支持 get 和 post 请求

编码：客户端 ：不需要额外做任何工作

服务器端：只需由服务器发送一个响应标头即可

存在浏览器兼容的问题

7. Ajax 交换模型？同步和异步区别？

Ajax 交换模型

1. 创建 XMLHttpRequest 对象

var request = createReq();

2. 设置监听回调 readyState 和 status 服务器响应

```
request.onreadystatechange = function () {
    if(request.readyState==4 && request.status==200) {
        var result = request.responseText;
        alert(result);
     }
};
```

3. 打开链接：

```
request.open('GET', '/node_ajax/test/get?username='+username);
```

4. 发送请求：GET 请求 POST 请求

```
request.send();
```

```
//创建发送 ajax 请求的 XMLHttpRequest 对象
 function createReq() {
    var httpReq = null;
    if(window.XMLHttpRequest) {
        httpReq = new XMLHttpRequest();
    } else {//IE6/5
    httpReq = new ActiveXObject("Microsoft.XMLHTTP");
    }
    return httpReq;
 }
```

文字描述：

用户发出异步请求；

创建 XMLHttpRequest 对象；

告诉 XMLHttpRequest 对象哪个函数会处理 XMLHttpRequest 对象状态的改变，为此要把对象的 onReadyStateChange 属性设置为响应该事件的 JavaScript 函数的引用；

创建请求，用 open 方法指定是 get 还是 post ，是否异步， url 地址；

发送请求， send 方法

接收结果并分析

实现刷新

设置

1). request.open(method, url, async)

2). 第 3 个参数就是用来指定是否异步，默认是 true(异步)，设置为 false 代表同步

同步和异步区别

1). 执行: request.send()

2). 异步: 此方法立即返回，后面立即获取请求结果数据得不到，

只能在监听回调中获取(当结果数据返回后回调)

3). 同步: 此方法不会立即返回，只有在服务器返回结果才返回，

在后面可以直接获取返回的结果数据，没有必要再设置监听回调

8. 同步和异步的区别?

同步：阻塞的

-张三叫李四去吃饭，李四一直忙得不停，张三一直等着，直到李四忙完两个人一块去吃饭

=浏览器向服务器请求数据，服务器比较忙，浏览器一直等着（页面白屏），直到服务器返回数据，浏览器才能显示页面

异步：非阻塞的

-张三叫李四去吃饭，李四在忙，张三说了一声然后自己就去吃饭了，李四忙完后自己去吃

=浏览器向服务器请求数据，服务器比较忙，浏览器可以自如的干原来的事情（显示页面），服务器返回数据的时候通知浏览器一声，浏览器把返回的数据再渲染到页面，局部更新

9. 如何解决跨域问题?

理解跨域的概念：协议、域名、端口都相同才同域，否则都是跨域

出于安全考虑，服务器不允许 ajax 跨域获取数据，但是可以跨域获取文件内容，所以基于这一点，可以动态创建 script 标签，使用标签的 src 属性访问 js 文件的形式获取 js 脚本，并且这个 js 脚本中的内容是函数调用，该函数调用的参数是服务器返回的数据，为了获取这里的参数数据，需要事先在页面中定义回调函数，在回调函数中处理服务器返回的数据，这就是解决跨域问题的主流解决方案

10. 页面编码和被请求的资源编码如果不一致如何处理？

对于 ajax 请求传递的参数，如果是 get 请求方式，参数如果传递中文，在有些浏览器会乱码，不同的浏览器对参数编码的处理方式不同，所以对于 get 请求的参数需要使用 encodeURIComponent 函数对参数进行编码处理，后台开发语言都有相应的解码 api。对于 post 请求不需要进行编码

11. 简述 ajax 的过程。

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

12、阐述一下异步加载。

1. 异步加载的方案： 动态插入 script 标签
2. 通过 ajax 去获取 js 代码，然后通过 eval 执行
3. script 标签上添加 defer 或者 async 属性
4. 创建并插入 iframe，让它异步执行 js

13. GET 和 POST 的区别，何时使用 POST？

GET：一般用于信息获取，使用 URL 传递参数，对所发送信息的数量也有限制，一般在 2000 个字符，有的浏览器是 8000 个字符

POST：一般用于修改服务器上的资源，对所发送的信息没有限制

在以下情况中，请使用 POST 请求：

1. 无法使用缓存文件（更新服务器上的文件或数据库）
2. 向服务器发送大量数据（POST 没有数据量限制）
3. 发送包含未知字符的用户输入时，POST 比 GET 更稳定也更可靠

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

1. 通过异步模式，提升了用户体验
2. 优化了浏览器和服务器之间的传输，减少不必要的数据往返，减少了带宽占用
3. Ajax 在客户端运行，承担了一部分本来由服务器承担的工作，减少了大用户量下的服务器负载。

15. Ajax 的最大的特点是什么。

Ajax 可以实现异步通信效果，实现页面局部刷新，带来更好的用户体验；按需获取数据，节约带宽资源；

16. ajax 的缺点

1、ajax 不支持浏览器 back 按钮。

2、安全问题 AJAX 暴露了与服务器交互的细节。

3、对搜索引擎的支持比较弱。

4、破坏了程序的异常机制。

17.ajax 请求的时候 get 和 post 方式的区别

get 一般用来进行查询操作，url 地址有长度限制，请求的参数都暴露在 url 地址当中，如果传递中文参数，需要自己进行编码操作，安全性较低。

post 请求方式主要用来提交数据，没有数据长度的限制，提交的数据内容存在于 http 请求体中，数据不会暴漏在 url 地址中。

18.解释 jsonp 的原理，以及为什么不是真正的 ajax

Jsonp 并不是一种数据格式，而 json 是一种数据格式，jsonp 是用来解决跨域获取数据的一种解决方案，具体是通过动态创建 script 标签，然后通过标签的 src 属性获取 js 文件中的 js 脚本，该脚本的内容是一个函数调用，参数就是服务器返回的数据，为了处理这些返回的数据，需要事先在页面定义好回调函数，本质上使用的并不是 ajax 技术

19.什么是 Ajax 和 JSON，它们的优缺点。

Ajax 是全称是 asynchronous JavaScript andXML，即异步 JavaScript 和 xml，用于在 Web 页面中实现异步数据交互，实现页面局部刷新。

优点：可以使得页面不重载全部内容的情况下加载局部内容，降低数据传输量，避免用户不断刷新或者跳转页面，提高用户体验

缺点：不能回退, 对搜索引擎不友好；要实现 ajax 下的前后退功能成本较大；可能造成请求数的增加跨域问题限制；

JSON 是一种轻量级的数据交换格式，ECMA 的一个子集

优点：轻量级、易于人的阅读和编写，便于机器（JavaScript）解析，支持复合数据类型（数组、对象、字符串、数字）

20.http 常见的状态码有那些？分别代表是什么意思？

200 - 请求成功

301 - 资源（网页等）被永久转移到其它 URL

404 - 请求的资源（网页等）不存在

500 - 内部服务器错误

21. 一个页面从输入 URL 到页面加载显示完成，这个过程中都发生了什么？

分为 4 个步骤：

1. 当发送一个 URL 请求时，不管这个 URL 是 Web 页面的 URL 还是 Web 页面上每个资源的 URL，浏览器都会开启一个线程来处理这个请求，同时在远程 DNS 服务器上启动一个 DNS 查询。这能使浏览器获得请求对应的 IP 地址。

2. 浏览器与远程 Web 服务器通过 TCP 三次握手协商来建立一个 TCP/IP 连接。该握手包括一个同步报文，一个同步-应答报文和一个应答报文，这三个报文在 浏览器和服务器之间传递。该握手首先由客户端尝试建立起通信，而后服务器应答并接受客户端的请求，最后由客户端发出该请求已经被接受的报文。

3. 一旦 TCP/IP 连接建立，浏览器会通过该连接向远程服务器发送 HTTP 的 GET 请求。远程服务器找到资源并使用 HTTP 响应返回该资源，值为 200 的 HTTP 响应状态表示一个正确的响应。

4. 此时，Web 服务器提供资源服务，客户端开始下载资源。

22. ajax 请求时，如何解释 json 数据

使用 eval() 或者 JSON.parse() 鉴于安全性考虑，推荐使用 JSON.parse()更靠谱，对数据的安全性更好。

23. javascript 的本地对象，内置对象和宿主对象

本地对象为独立于宿主环境的 ECMAScript 提供的对象，包括 Array Object RegExp 等可以 new 实例化的对象

内置对象为 Gload，Math 等不可以实例化的(他们也是本地对象，内置对象是本地对象的一个子集)

宿主对象为所有的非本地对象，所有的 BOM 和 DOM 对象都是宿主对象，如浏览器自带的 document,window 等对象

24. 为什么利用多个域名来存储网站资源会更有效？

确保用户在不同地区能用最快的速度打开网站，其中某个域名崩溃用户也能通过其他郁闷访问网站，并且不同的资源放到不同的服务器上有利于减轻单台服务器的压力。

25. 请说出三种减低页面加载时间的方法

1、压缩 css、js 文件

2、合并 js、css 文件，减少 http 请求

3、外部 js、css 文件放在最底下

4、减少 dom 操作，尽可能用变量替代不必要的 dom 操作

26. HTTP 状态码都有那些。

200 OK //客户端请求成功

400 Bad Request //客户端请求有语法错误，不能被服务器所理解

403 Forbidden //服务器收到请求，但是拒绝提供服务

404 Not Found //请求资源不存在，输入了错误的 URL

500 Internal Server Error //服务器发生不可预期的错误

五、JS 高级

1. Javascript 如何实现继承？

1、构造继承

2、原型继承

3、实例继承

4、拷贝继承

原型 prototype 机制或 apply 和 call 方法去实现较简单，建议使用构造函数与原型混合方式。

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

function Child(){
    this.age = 28;
}
Child.prototype = new Parent();//继承了 Parent，通过原型
var demo = new Child();
alert(demo.age);
alert(demo.name);//得到被继承的属性
```

2. Javascript 作用链域?

全局函数无法查看局部函数的内部细节，但局部函数可以查看其上层的函数细节，直至全局细节。

当需要从局部函数查找某一属性或方法时，如果当前作用域没有找到，就会上溯到上层作用域查找，

直至全局函数，这种组织形式就是作用域链。

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

parseInt() 函数能解析一个字符串，并返回一个整数，需要两个参数 (val, radix)，

其中 radix 表示要解析的数字的基数。【该值介于 2 ~ 36 之间，并且字符串中的数字不能大于 radix 才能正确返回数字结果值】;

但此处 map 传了 3 个 (element, index, array),我们重写 parseInt 函数测试一下是否符合上面的规则。

```
function parseInt(str, radix) {
    return str+'-'+radix;
};
var a=["1", "2", "3"];
a.map(parseInt);   // ["1-0", "2-1", "3-2"]  不能大于 radix
```

因为二进制里面，没有数字 3,导致出现超范围的 radix 赋值和不合法的进制解析，才会返回 NaN

所以["1", "2", "3"].map(parseInt) 答案也就是：[1, NaN, NaN]

4. 什么是闭包（closure），为什么要用它？

闭包是指有权访问另一个函数作用域中变量的函数，创建闭包的最常见的方式就是在一个函数内创建另一个函数，通过另一个函数访问这个函数的局部变量,利用闭包可以突破作用链域，将函数内部的变量和方法传递到外部。

闭包的特性：

1.函数内再嵌套函数

2.内部函数可以引用外层的参数和变量

3.参数和变量不会被垃圾回收机制回收

```
//li 节点的 onclick 事件都能正确的弹出当前被点击的 li 索引
 <ul id="testUL">
    <li> index = 0</li>
    <li> index = 1</li>
    <li> index = 2</li>
    <li> index = 3</li>
</ul>
```

```
<script type="text/javascript">
        var nodes = document.getElementsByTagName("li");
        for(i = 0;i<nodes.length;i+= 1){
                nodes[i].onclick = (function(i){
                                return function() {
                                    console.log(i);
                                } //不用闭包的话，值每次都是 4
                            })(i);
        }
</script>
```

执行 say667()后,say667()闭包内部变量会存在,而闭包内部函数的内部变量不会存在

使得 Javascript 的垃圾回收机制 GC 不会收回 say667()所占用的资源

因为 say667()的内部函数的执行需要依赖 say667()中的变量

这是对闭包作用的非常直白的描述

```
   function say667() {
        // Local variable that ends up within closure
        var num = 666;
        var sayAlert = function() {
                alert(num);
        }
        num++;
        return sayAlert;
}

  var sayAlert = say667();
```

sayAlert()//执行结果应该弹出的 667

5. javascript 代码中的"use strict";是什么意思 ？使用它区别是什么？

use strict 是一种 ECMAscript 5 添加的（严格）运行模式,这种模式使得 Javascript 在更严格的条件下运行,

使 JS 编码更加规范化的模式,消除 Javascript 语法的一些不合理、不严谨之处，减少一些怪异行为。

默认支持的糟糕特性都会被禁用，比如不能用 with，也不能在意外的情况下给全局变量赋值;

全局变量的显示声明,函数必须声明在顶层，不允许在非函数代码块内声明函数,arguments.callee 也不允许使用；

消除代码运行的一些不安全之处，保证代码运行的安全,限制函数中的 arguments 修改，严格模式下的 eval 函数的行为和非严格模式的也不相同;

提高编译器效率，增加运行速度；

为未来新版本的 Javascript 标准化做铺垫。

6. 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

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

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

区别：

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

2. CMD 推崇依赖就近，AMD 推崇依赖前置。看代码：

```
// CMD
define(function(require, exports, module) {
    var a = require('./a')
    a.doSomething()
    // 此处略去 100 行
    var b = require('./b') // 依赖可以就近书写
    b.doSomething()
    // ...
})

// AMD 默认推荐
define(['./a', './b'], function(a, b) { // 依赖必须一开始就写好
    a.doSomething()
    // 此处略去 100 行
    b.doSomething()
    // ...
})
```

8. 异步加载 JS 的方式有哪些？

(1) defer，只支持 IE

(2) async：

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

9. jQuery 的 slideUp 动画 ，如果目标元素是被外部事件驱动，当鼠标快速地连续触发外部元素事件，动画会滞后的反复执行，该如何处理呢？

jquery stop(): 如：$("#div").stop().animate({width:"100px"},100);

10. JQuery 一个对象可以同时绑定多个事件，这是如何实现的？

多个事件同一个函数：

```
$("div").on("click mouseover", function(){});
```

多个事件不同函数

```
$("div").on({
    click: function(){},
    mouseover: function(){}
});
```

11. 使用 JS 实现获取文件扩展名？

```
function getFileExtension(filename) {
  return filename.slice((filename.lastIndexOf(".") - 1 >>> 0) + 2);
}
```

String.lastIndexOf() 方法返回指定值（本例中的'.'）在调用该方法的字符串中最后出现的位置，如果没找到则返回 -1。

对于'filename'和'.hiddenfile'，lastIndexOf 的返回值分别为 0 和-1 无符号右移操作符(»>) 将-1 转换为 4294967295，将-2 转换为 4294967294，这个方法可以保证边缘情况时文件名不变。

String.prototype.slice() 从上面计算的索引处提取文件的扩展名。如果索引比文件名的长度大，结果为""。

12.ECMAScript6 相关

Object.is() 与原来的比较操作符“ ===”、“ ==”的区别？

两等号判等，会在比较时进行类型转换；

三等号判等(判断严格)，比较时不进行隐式类型转换,（类型不同则会返回 false）；

Object.is 在三等号判等的基础上特别处理了 NaN 、-0 和 +0 ，保证 -0 和 +0 不再相同，

但 Object.is(NaN, NaN) 会返回 true.

Object.is 应被认为有其特殊的用途，而不能用它认为它比其它的相等对比更宽松或严格。

13.页面重构怎么操作？

网站重构：在不改变外部行为的前提下，简化结构、添加可读性，而在网站前端保持一致的行为。

也就是说是在不改变 UI 的情况下，对网站进行优化，在扩展的同时保持一致的 UI。

对于传统的网站来说重构通常是：

表格(table)布局改为 DIV+CSS

使网站前端兼容于现代浏览器(针对于不合规范的 CSS、如对 IE6 有效的)

对于移动平台的优化

针对于 SEO 进行优化

深层次的网站重构应该考虑的方面

减少代码间的耦合

让代码保持弹性

严格按规范编写代码

设计可扩展的 API

代替旧有的框架、语言(如 VB)

增强用户体验

通常来说对于速度的优化也包含在重构中

压缩 JS、CSS、image 等前端资源(通常是由服务器来解决)

程序的性能优化(如数据读写)

采用 CDN 来加速资源加载

对于 JS DOM 的优化

HTTP 服务器的文件缓存

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

注：这题胜在区分度高，知识点覆盖广，再不懂的人，也能答出几句，

而高手可以根据自己擅长的领域自由发挥，从 URL 规范、HTTP 协议、DNS、CDN、数据库查询、

到浏览器流式解析、CSS 规则构建、layout、paint、onload/domready、JS 执行、JS API 绑定等等；

详细版：

1、浏览器会开启一个线程来处理这个请求，对 URL 分析判断如果是 http 协议就按照 Web 方式来处理;

2、调用浏览器内核中的对应方法，比如 WebView 中的 loadUrl 方法;

3、通过 DNS 解析获取网址的 IP 地址，设置 UA 等信息发出第二个 GET 请求;

4、进行 HTTP 协议会话，客户端发送报头(请求报头);

5、进入到 web 服务器上的 Web Server，如 Apache、Tomcat、Node.JS 等服务器;

6、进入部署好的后端应用，如 PHP、Java、JavaScript、Python 等，找到对应的请求处理;

7、处理结束回馈报头，此处如果浏览器访问过，缓存上有对应资源，会与服务器最后修改时间对比，一致则返回 304;

8、浏览器开始下载 html 文档(响应报头，状态码 200)，同时使用缓存;

9、文档树建立，根据标记请求所需指定 MIME 类型的文件（比如 css、js）,同时设置了 cookie;

10、页面开始渲染 DOM，JS 根据 DOM API 操作 DOM,执行事件绑定等，页面显示完成。

简洁版：

浏览器根据请求的 URL 交给 DNS 域名解析，找到真实 IP，向服务器发起请求；

服务器交给后台处理完成后返回数据，浏览器接收文件（HTML、JS、CSS、图象等）；

浏览器对加载到的资源（HTML、JS、CSS 等）进行语法解析，建立相应的内部数据结构（如 HTML 的 DOM）；

载入解析到的资源文件，渲染页面，完成。

15. 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

16. 简述列举文档对象模型 DOM 里 document 的常用的查找访问节点的方法，并做简单说明

document.getElementById 根据元素 id 查找元素

document.getElementByName 根据元素 name 查找元素

document.getElementTagName 根据指定的元素名查找元素

document.getElementsByClassName()根据元素 class 查找元素,有兼容问题（ie8 及其以下不支持）

h5 中：

document.querySelector()

document.querySelectorAll(),返回伪数组

出现 dom 元素发生改变（重排重绘现象），不能使用

17. 重排重绘

重绘是一个元素外观的改变所触发的浏览器行为，例如改变 outline、背景色等属性。浏览器会根据元素的新属性重新绘制，

使元素呈现新的外观。重绘不会带来重新布局，所以并不一定伴随重排。

渲染对象在创建完成并添加到渲染树时，并不包含位置和大小信息。计算这些值的过程称为布局或重排

"重绘"不一定需要"重排"，比如改变某个网页元素的颜色，就只会触发"重绘"，不会触发"重排"，因为布局没有改变。

但是，"重排"必然导致"重绘"，比如改变一个网页元素的位置，就会同时触发"重排"和"重绘"，因为布局改变了。

常见的触发重排的操作

重排的成本比重绘的成本高得多的多。DOM Tree 里的每个结点都会有 重排方法，

一个结点的重排很有可能导致子结点，甚至父点以及同级结点的 重排。在一些高性能的电脑上也许还没什么，

但是如果重排发生在手机上，那么这个过程是非常痛苦和耗电的。

所以，下面这些动作有很大可能会是成本比较高的。

当你增加、删除、修改 DOM 结点时，会导致 Reflow , Repaint。

当你移动 DOM 的位置

当你修改 CSS 样式的时候。

当你 Resize 窗口的时候（移动端没有这个问题）

当你修改网页的默认字体时。

获取某些属性时

注：display:none 会触发 reflow，而 visibility:hidden 只会触发 repaint，因为没有发现位置变化。

18. 判断一个字符串中出现次数最多的字符，并统计次数

```
var str = 'shdshdfjkfjfdgjkjdksgjskdjfsfsfsfjksjkfdkjf'
var arr = str.split('').sort()//把字符串变为数组，在排序
str = arr.join('') //把排序好的数组，变为字符串

var count = 0    //不能使用 undefined，转化为数字是 NaN
var char = 0

var reg = /(\w)\1+/g
str.replace(reg,function(parent,son){
    if(parent.length >count){
        count = parent.length
        char = son
```

```
    }
})
console.log("最多的字符为:"+char+";个数为:"+count)
```

拓展：replace 方法：

replace 语法：

```
replace(reg,function(parent,son1,son2,index){
    第一个参数:匹配成功的子字符串(parent)
    第二个参数:第一个分组内容;(son)
    第三个参数:第二个分组内容;(son2)
    ...
    最后一个为开始的下标
})
```

```
<script type="text/javascript">
    var data = '2016-6-8'
    var reg = /(\d*)-/g
    var data = data.replace(reg,function(parent,son1,index){    //此时没有 son2
        console.log(parent) //2016- , 6-
        console.log(son1) //2016 , 6
        console.log(index) //0    5    开始匹配的下标
        return son1+'.'
    })
    console.log(data) //2016.6.8
</script>
```

使用 replace()基本方法

```
var data = '2016-6-8'
```

```
var reg = /-/g
var data = data.replace(reg,'.')
console.log(data)    //2016.6.8
```

19. 查看下面代码

```
var nodes=document.getElementByTagName('button')
var len=nodes.length
for(var i=0;i<len;i++){
    nodes[i].addEventListener('click',function(){
        console.log('i='+i)
    })
}
```

请问用户点击第一个和第二个按钮控制台会输出什么　都是 i=len 的值

解决方案：

方法一：使用 let 解决(let 有兼容性,ES6)

```
var a = document.getElementsByTagName('a')
for (let i= 0;i<a.length;i++) {
    a[i].onclick = function (){
        a[i].parentNode.style.display = 'none' //不会影响 dom 结构，节约浏览器开销
    }
}
```

方法二：使用 this 解决

```
var a = document.getElementsByTagName('a')
for (let i= 0;i<a.length;i++) {
```

```
        a[i].onclick = function(){
            this.parentNode.style.display = 'none'
        }
    }
```

方法三：立即执行函数表达式 IIFE，闭包原理

```
    var a = document.getElementsByTagName('a')
    for (let i= 0;i<a.length;i++) {
        (function(j){
            a[j].onclick = function(){
                a[j].parentNode.style.display='none'
            }
        })(i)
    }
```

拓展：

let，var，const

let 关键字

作用:与 var 类似，用于声明一个变量

特点:1 只在块作用域内有效 2 不能重复声明 3 不会预处理，不存在提升

应用:

循环遍历加监听

```
var btns = document.getElementsByTagName('button');
for (let i = 0,length=btns.length; i < length; i++) {
   btns[i].onclick = function () {
       alert(i);
   }
}
```

在非函数块中声明变量, 应该用 let （for 语句 ，if 语句）

const 关键字

作用:定义一个常量 （一般用大写来保存）

特点:1 不能修改 2 也是块作用域有效（1 不能重复声明 2 不会预处理, 不存在提升） --let 的特点也符合

应用:保存应用需要的常量数据

```
const URL = 'http://www.atguigu.com';
```

this 的几种情况:

1.当以函数的形式调用时，this 永远都是 window

2.以方法的形式调用时，this 就是调用方法的那个对象

3.以构造函数的形式调用，this 就是新创建的那个对象

4.回调函数: 看背后是通过谁来调用的: window/其它

回调函数中的 this 不能确定，比如定时器，有时会有事件的绑定 onclick,事件操作才会执行

5.当使用 call 或 apply 去调用一个函数时，this 就是他们的第一个参数(如果没有参数就是 window，因为是函数调用)

区别 bind()与 call()和 apply()?

bind()将函数内的 this 绑定为 obj, 并将函数返回

call(obj)和 apply(obj):将函数内部的 this 绑定为 obj,调用函数

call(obj, a, b)和 apply(obj,[a ,b]):传递参数的方式不一样

立即执行函数表达式 : IIFE

```
(function(window, obj){
    //函数体
})(window, obj)
```

jQuery 就是使用 IIFE 实现的：

```
(function(window){
    var jQuery = function(obj) {
        return {
            val : function(){}
        }
    }

    jQuery.get = function(){}

    window.jQuery = window.$ = jQuery

})(window)
```

解释：参数传入 window 就可以直接使用 jQuery

把 jQuery 赋值给 window.jQuery = window.$ = jQuery，就可以直接用$,但是如果有其他的函数使用了$,就必须使用 jQuery

obj 形参可以是任意类型：

1，选择器 （标签和字符串选择器，jQuery 中就会通过正则表达式来区分）

选择器：有特定格式的字符串，作用，用于查找 dom 元素

2，函数 function:参考 window.onload 给整个文档加了一个监听

3，对象（jQuery 会对参数进行判断，针对不同的参数对应不同的事件，依次来实现的）

20. 请写出以下输出结果：

```
function Foo() {
    getName = function () { alert (1); };
    return this;
```

```
}
Foo.getName = function () { alert (2);};
Foo.prototype.getName = function () { alert (3);};
var getName = function () { alert (4);};
function getName() { alert (5);}

//请写出以下输出结果：
Foo.getName();
getName();
Foo().getName();
getName();
new Foo.getName();
new Foo().getName();
new new Foo().getName();
```

答案：（2,4,1,1,2,3,3）

21.请写出以下输出结果：

```
function f(y) {
    var x=y*y;
    return x;
}

for(x=0;x< 5;x++) {
    y=f(x);
    document.writeln(y);
}
```

答案：0 1 4 9 16

22.以下代码输出的内容

```
var User = {
    count :1,
    getCount : function(){
        return this.count;
    }
}
console.log(User.getCount())
var func = User.getCount;
console.log(func())
```

答案:1 undefined

23.jquery.extend 与 jquery.fn.extend 的区别？

jquery.extend 为 jquery 类添加类方法，可以理解为添加静态方法

jquery.fn.extend:

源码中 jquery.fn = jquery.prototype，所以对 jquery.fn 的扩展，就是为 jquery 类添加成员函数

使用：

jquery.extend 扩展，需要通过 jquery 类来调用，而 jquery.fn.extend 扩展，所有 jquery 实例都可以直接调用。

24.Jquery 与 jQuery UI 有啥区别？

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

jQuery UI 则是在 jQuery 的基础上，利用 jQuery 的扩展性，设计的插件。

提供了一些常用的界面元素，诸如对话框、拖动行为、改变大小行为等等

25. jquery 中如何将数组转化为 json 字符串，然后再转化回来？

jQuery 中没有提供这个功能，所以你需要先编写两个 jQuery 的扩展：

```
 $.fn.stringifyArray = function(array) {
      return JSON.stringify(array)
}

$.fn.parseArray = function(array) {
      return JSON.parse(array)
}
```

然后调用：

```
$("").stringifyArray(array)
```

26. 针对 jQuery 的优化方法？

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

频繁操作的 DOM，先缓存起来再操作。用 Jquery 的链式调用更好。

比如：var str=$("a").attr("href");

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

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

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

27. JQuery 一个对象可以同时绑定多个事件，这是如何实现的？

jQuery 可以给一个对象同时绑定多个事件，低层实现方式是使用 addEventListner 或 attachEvent 兼容不同的浏览器实现事件的绑定，这样可以给同一个对象注册多个事件。

28. 知道什么是 webkit 么？知道怎么用浏览器的各种工具来调试和 debug 代码么？

Webkit 是浏览器引擎，包括 html 渲染和 js 解析功能，手机浏览器的主流内核，与之相对应的引擎有 Gecko（Mozilla Firefox 等使用）和 Trident（也称 MSHTML，IE 使用）。

对于浏览器的调试工具要熟练使用，主要是页面结构分析，后台请求信息查看，js 调试工具使用，熟练使用这些工具可以快速提高解决问题的效率

29. 我们给一个 dom 同时绑定两个点击事件，一个用捕获，一个用冒泡，你来说下会执行几次事件，然后会先执行冒泡还是捕获

对两种事件模型的理解

30. 如何消除一个数组里面重复的元素？

```
var arr=[1,2,3,3,4,4,5,5,6,1,9,3,25,4];
        function deRepeat(){
            var newArr=[];
            var obj={};
            var index=0;
            var l=arr.length;
            for(var i=0;i<l;i++){
                if(obj[arr[i]]==undefined)
                  {
                    obj[arr[i]]=1;
```

```
                newArr[index++]=arr[i];
            }
        else if(obj[arr[i]]==1)
            continue;
    }
    return newArr;
}
var newArr2=deRepeat(arr);
alert(newArr2); //输出 1,2,3,4,5,6,9,25
```

31.小贤是一条可爱的小狗(Dog)，它的叫声很好听(wow)，每次看到主人的时候就会乖乖叫一声(yelp)。从这段描述可以得到以下对象：

```
function Dog() {
    this.wow = function() {
        alert('Wow');
    }
    this.yelp = function() {
        this.wow();
    }
}
```

小芒和小贤一样，原来也是一条可爱的小狗，可是突然有一天疯了(MadDog)，一看到人就会每隔半秒叫一声(wow)地不停叫唤(yelp)。请根据描述，按示例的形式用代码来实。（继承，原型，setInterval）

```
function MadDog() {
  this.yelp = function() {
    var self = this;
    setInterval(function() {
      self.wow();
```

```
        }, 500);
    }
}
MadDog.prototype = new Dog();
//for test
var dog = new Dog();
dog.yelp();
var madDog = new MadDog();
madDog.yelp();
```

32. 下面这个 ul，如何点击每一列的时候 alert 其 index?（闭包）

```
<ul id=”test”>
    <li>这是第一条</li>
    <li>这是第二条</li>
    <li>这是第三条</li>
</ul>


    // 方法一：
var lis=document.getElementById('2223').getElementsByTagName('li');
for(var i=0;i<3;i++)
{
  lis[i].index=i;
  lis[i].onclick=function(){
    alert(this.index);
  };
}
```

```
//方法二：
var lis=document.getElementById('2223').getElementsByTagName('li');
for(var i=0;i<3;i++){
   lis[i].index=i;
   lis[i].onclick=(function(a){
      return function() {
         alert(a);
      }
   })(i);
}
```

33. 请评价以下代码并给出改进意见。

```
if(window.addEventListener){
   var addListener = function(el,type,listener,useCapture){
      el.addEventListener(type,listener,useCapture);
  };
} else if(document.all){
   addListener = function(el,type,listener){
      el.attachEvent("on"+type,function(){
        listener.apply(el);
     });
  }
}
```

不应该在 if 和 else 语句中声明 addListener 函数，应该先声明；

不需要使用 window.addEventListener 或 document.all 来进行检测浏览器，应该使用能力检测；

由于 attachEvent 在 IE 中有 this 指向问题，所以调用它时需要处理一下

改进如下：

```
function addEvent(elem, type, handler){
    if(elem.addEventListener){
        elem.addEventListener(type, handler, false);
    }else if(elem.attachEvent){
        elem['temp' + type + handler] = handler;
        elem[type + handler] = function(){
        elem['temp' + type + handler].apply(elem);
    };
    elem.attachEvent('on' + type, elem[type + handler]);
 }else{
    elem['on' + type] = handler;
    }
}
```

34. 在 Javascript 中什么是伪数组？如何将伪数组转化为标准数组？

伪数组（类数组）：无法直接调用数组方法或期望 length 属性有什么特殊的行为，但仍可以对真正数组遍历方法来遍历它们。典型的是函数的 argument 参数，还有像调用 getElementsByTagName,document.childNodes 之类的,它们都返回 NodeList 对象都属于伪数组。可以使用 Array.prototype.slice.call(fakeArray)将数组转化为真正的 Array 对象。

假设接第八题题干，我们要给每个 log 方法添加一个”(app)”前缀，比如’hello world!’ ->’(app)hello world!’。方法如下：

```
function log(){
    var args = Array.prototype.slice.call(arguments);  //为了使用 unshift 数组方法，将 argument 转化为真正的数组
    args.unshift('(app)');
    console.log.apply(console, args);
```

```
  };
```

35.对作用域上下文和 this 的理解，看下列代码：

```
var User = {
 count: 1,
 getCount: function() {
   return this.count;
 }
};
console.log(User.getCount());  // what?
var func = User.getCount;
console.log(func());  // what?
```

问两处 console 输出什么？为什么？

答案是 1 和 undefined。

func 是在 winodw 的上下文中被执行的，所以会访问不到 count 属性。

继续追问，那么如何确保 Uesr 总是能访问到 func 的上下文，即正确返回 1。正确的方法是使用 Function.prototype.bind。兼容各个浏览器完整代码如下：

```
Function.prototype.bind = Function.prototype.bind || function(context){
  var self = this;
  return function(){
    return self.apply(context, arguments);
  };
}
var func = User.getCount.bind(User);
console.log(func());
```

36. Javascript 作用链域?

理解变量和函数的访问范围和生命周期，全局作用域与局部作用域的区别，JavaScript 中没有块作用域，函数的嵌套形成不同层次的作用域，嵌套的层次形成链式形式，通过作用域链查找属性的规则需要深入理解。

37. 谈谈 This 对象的理解。

理解不同形式的函数调用方式下的 this 指向，理解事件函数、定时函数中的 this 指向，函数的调用形式决定了 this 的指向。

38. 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 个步骤，一次解析成 js 语句，一次执行）

39. 什么是闭包（closure），为什么要用它？

简单的理解是函数的嵌套形成闭包，闭包包括函数本身已经它的外部作用域

使用闭包可以形成独立的空间，延长变量的生命周期，报存中间状态值

40. javascript 代码中的"use strict";是什么意思 ？使用它区别是什么？

意思是使用严格模式，使用严格模式，一些不规范的语法将不再支持

41. 如何判断一个对象是否属于某个类？

instanceof

42. new 操作符具体干了什么呢?

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

2、属性和方法被加入到 this 引用的对象中。

3、新创建的对象由 this 所引用，并且最后隐式的返回 this

43. 用原生 JavaScript 的实现过什么功能吗？

主要考察原生 js 的实践经验

44. 数组和对象有哪些原生方法，列举一下？

Array.concat( ) 连接数组

Array.join( ) 将数组元素连接起来以构建一个字符串

Array.length 数组的大小

Array.pop( ) 删除并返回数组的最后一个元素

Array.push( ) 给数组添加元素

Array.reverse( ) 颠倒数组中元素的顺序

Array.shift( ) 将元素移出数组

Array.slice( ) 返回数组的一部分

Array.sort( ) 对数组元素进行排序

Array.splice( ) 插入、删除或替换数组的元素

Array.toLocaleString( ) 把数组转换成局部字符串

Array.toString( ) 将数组转换成一个字符串

Array.unshift( ) 在数组头部插入一个元素

Object.hasOwnProperty( ) 检查属性是否被继承

Object.isPrototypeOf( ) 一个对象是否是另一个对象的原型

Object.propertyIsEnumerable( ) 是否可以通过 for/in 循环看到属性

Object.toLocaleString( ) 返回对象的本地字符串表示

Object.toString( ) 定义一个对象的字符串表示

Object.valueOf( ) 指定对象的原始值

45. JS 怎么实现一个类。怎么实例化这个类

严格来讲 js 中并没有类的概念，不过 js 中的函数可以作为构造函数来使用，通过 new 来实例化，其实函数本身也是一个对象。

46. JavaScript 中的作用域与变量声明提升？

理解 JavaScript 的预解析机制，js 的运行主要分两个阶段：js 的预解析和运行，预解析阶段所有的变量声明和函数定义都会提前，但是变量的赋值不会提前

47. javascript 对象的几种创建方式？

1. 工厂模式
2. 构造函数模式
3. 原型模式
4. 混合构造函数和原型模式
5. 动态原型模式
6. 寄生构造函数模式
7. 稳妥构造函数模式

48. javascript 继承的 6 种方法？

1. 原型链继承
2. 借用构造函数继承
3. 组合继承(原型+借用构造)
4. 原型式继承
5. 寄生式继承
6. 寄生组合式继承

49. eval 是做什么的？

1. 它的功能是把对应的字符串解析成 JS 代码并运行

2. 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）

50. JavaScript 原型，原型链 ？有什么特点？

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

2. 原型链是由一些用来继承和共享属性的对象组成的（有限的）对象链

51. 简述一下 Sass、Less，且说明区别？

他们是动态的样式语言，是 CSS 预处理器,CSS 上的一种抽象层。他们是一种特殊的语法/语言而编译成 CSS。

变量符不一样，less 是@，而 Sass 是$;

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

Sass 是基于 Ruby 的，是在服务端处理的，而 Less 是需要引入 less.js 来处理 Less 代码输出 Css 到浏览器

52. 关于 javascript 中 apply()和 call()方法的区别？

相同点:两个方法产生的作用是完全一样的

不同点:方法传递的参数不同

Object.call(this,obj1,obj2,obj3)

Object.apply(this,arguments)

apply()接收两个参数，一个是函数运行的作用域(this)，另一个是参数数组。

call()方法第一个参数与 apply()方法相同，但传递给函数的参数必须列举出来。

53. 说说你对 this 的理解？

在 JavaScript 中，this 通常指向的是我们正在执行的函数本身，或者是，指向该函数所属的对象。

全局的 this → 指向的是 Window

函数中的 this → 指向的是函数所在的对象

对象中的 this → 指向其本身

54. 事件委托是什么？

让利用事件冒泡的原理，让自己的所触发的事件，让他的父元素代替执行！

55. 谈一下 JS 中的递归函数，并且用递归简单实现阶乘？

递归即是程序在执行过程中不断调用自身的编程技巧，当然也必须要有一个明确的结束条件，不然就会陷入死循环。

56. 请用正则表达式写一个简单的邮箱验证。

/^[a-zA-Z0-9_-]+@[a-zA-Z0-9_-]+(\.[a-zA-Z0-9_-]+)+$/;

57. 简述一下你对 web 性能优化的方案？

1、尽量减少 HTTP 请求

2、使用浏览器缓存

3、使用压缩组件

4、图片、JS 的预载入

5、将脚本放在底部

6、将样式文件放在页面顶部

7、使用外部的 JS 和 CSS

8、精简代码

58.在 JS 中有哪些会被隐式转换为 false

Undefined、null、关键字 false、NaN、零、空字符串

59.定时器 setInterval 有一个有名函数 fn1，setInterval（fn1,500）与 setInterval（fn1(),500）有什么区别？

第一个是重复执行每 500 毫秒执行一次，后面一个只执行一次。

60.外部 JS 文件出现中文字符，会出现什么问题，怎么解决？

会出现乱码，加 charset=”utf-8”；

61.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');
}
```

62.JSON 的了解

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

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

63. 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

64. 前端开发的优化问题（看雅虎 14 条性能优化原则）。

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

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

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

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

（5） 少用全局变量、缓存 DOM 节点查找的结果。减少 IO 读取操作。

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

（7） 图片预加载，将样式表放在顶部，将脚本放在底部 加上时间戳。

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

65. 函数的 3 种不同角色

* 一般函数 ： 直接调用
* 构造函数 ： 通过 new 调用
* 对象 ： 通过.调用内部的属性/方法

66. 如何判断函数中的 this

* 显式指定谁: obj.xxx()
* 通过 call/apply 指定谁调用: xxx.call(obj)
* 不指定谁调用: xxx() : window
* 回调函数: 看背后是通过谁来调用的: window/其它

67. 说说你对原型与原型链的理解

* 实例对象的__proto__(隐式原型)属性等于其构造函数的 prototype(显式原型)属性,默认为一个空 object 对象

* 所有的实例对象都有__proto__属性，原型对象是一个实例对象

* 这样通过__proto__属性就形成了一个链的结构---->原型链

* 当我们通过对象查找对象内部的属性/方法时，会沿着这个原型链查找

68. 说说你对变量提升和函数提升的理解

* 变量提升：在变量(var)定义语句之前，就可以访问到这个变量，值为 undefined

* 函数提升：在函数声明语句之前，直接就可以调用此函数，值为函数定义

* 在执行全局代码或调用函数执行函数体之前，会初始化 var 定义的变量和 function 定义的函数

69. 说说你对作用域与作用域链的理解

* 作用域:一块代码区域

* 分类:

    * 全局

    * 函数

    * js 没有块作用域(在 ES6 之前)

* 作用

    * 隔离变量，可以在不同作用域定义同名的变量不冲突

    * 查找变量

* 作用域链:

    * 多个嵌套的作用域形成的链，链的方向的由内向外(链头当前作用域，链尾是全局作用域)

    * 当我们在查找变量时，沿着作用链由由内向外查找

70. 说说你对事件处理机制的理解

* 把代码分为 2 类: 初始化代码和事件回调代码

* 浏览器有些管理模块: ajax 模块/定时器模块/DOM 事件模块运行在分线程

* 程序开始执行初始化代码, 可能: 设置监听,设置定时器, 发送 ajax, 将回调函数交给对应的模块进行管理

* 当事件发生或指定时间到达时, 管理模块会将回调添加到 callback Queue

* 当初始化代码执行完后, 才会遍历取出 callback Queue 中的回调函数依次调用

71. 说说你对栈和队列的理解

* 栈 :

  后进先出

  上下文环境栈

  行为:

    压栈(push)

    出栈(pop)

    查看栈顶元素(peek)

    清栈(clear)

    大小(size)

    判断是否为空(isEmpty)

* 队列:

  先进先出

  事件/消息队列

  行为:

    加入队列 : enqueue

    从队列移除: dequeue

查看第一个: first

清空队列: (clear)

大小(size)

判断是否为空(isEmpty)

72. 说说你对闭包的理解

* 理解:
    * 当嵌套的内部函数引用了外部函数的变量时就产生了闭包
    * 通过 chrome 工具得知: 闭包本质是内部函数中的一个对象, 这个对象中包含引用变量属性
* 作用:
    * 延长局部变量的生命周期
    * 让函数外部能操作内部的局部变量

73. 编码实现继承

```
function Parent(xxx){this.xxx = xxx}
Parent.prototype.test = function(){};
function Child(xxx,yyy){
    Parent.call(this, xxx);//借用构造函数    this.Parent(xxx)
}
Child.prototype = new Parent(); //得到 test()
var child = new Child();
```

74. 编码实现一个自定义的模块

```
(function(window){
    var msg = 'atguigu.com';
```

```
        function showMsg(){
            alert(msg)
        }
        window.myModule = {
            showMsg : showMsg
        };
    })(window)
```

75. 说说你对 jQuery 链式调用的理解

> 描述: 通过点可以不断调用方法

> 好处: 编码简洁

> 原因: 方法会返回 this(jQuery 对象)

76. 说说你对事件委托的理解

> - 过程:

> 1. 将多个子元素(li)的事件监听委托给父辈元素(ul)处理

> 2. 监听回调是加在了父辈元素上

> 3. 当操作任何一个子元素(li)时，事件会冒泡到父辈元素(ul)

> 4. 父辈元素不会直接处理事件，而是根据 event.target 得到发生事件的子元素(li)，通过这个子元素调用事件回调函数 >

> - 好处:

> 1. 减少事件监听的数量: n--->1

> 2. 添加新的子元素，自动有事件响应处理

77.window.onload 与$(document).ready() 区别

>1. 执行时间

> - window.onload 必须等到页面内包括图片的所有元素加载完毕后才能执行。

> - $(document).ready()是 DOM 结构加载完毕后就执行，不必等到图片加载完毕。

>2. 编写个数不同

> - window.onload 不能同时编写多个，如果有多个 window.onload 方法，只会执行一个

> - $(document).ready()可以同时编写多个，并且都可以得到执行

>3. 简化写法

> - window.onload 没有简化写法

> - $(document).ready(function(){})可以简写成$(function(){});

78. jQuery 核心函数的常用几种用法（能写几种写几种）

>* 作为一般函数调用 :$(params)

> * callback function : 绑定文档加载完成的回调

> * 选择器字符串 : 查找所有匹配的 dom 元素, 并包装为 jQuery 对象返回

> * 标签字符串 : 生成 dom 元素对象, 并包装为 jQuery 对象返回

> * dom 元素对象: 将指定的 dom 元素包装为 jQuery 对象返回 (用得较多的是 this)

>* 作用对象使用(工具方法)

> * $.each() : 遍历数组/对象

79. 什么是 jQuery 插件，如何编写 jQuery 插件（用到的函数是什么）

> jQuery 插件是地方编写的，基于 jQuery 基础上的，对 jQuery 功能的扩展

> 编写 jQuery 插件，有两个方法

>```

>$.extend({

>  leftTrim : function (str) {// '  abc  '--->'abc  '

>  return str.replace(/^\s+/, '');

>  },

>  rightTrim : function (str) {// '  abc  '--->'abc  '

>  return str.replace(/\s+$/, '');

>  }

>});

>```

>```

>$.fn.extend({

>  reverseCheck : function () {

>  this.each(function () {

>  this.checked = !this.checked;

>  });

>  }

>});

>```

80. 跨域是什么，jsop 解决跨域的原理

> 跨域就是页面中的 ajax 需要访问不在同一个域内的数据所产生的现象

> 同源策略： 协议，域名，端口号 必须都相同

> 解决跨域问题，有两种方法， jsop 和 CORS，

> jsop 的方法是利用 script 标签不受同源策略约束，可以自由发送和接受请求，服务器通过把数据放到返回的 js 中，作为函数调用的参数传递。

81.写出 jQuery 操作 ajax 的几种方法（能写几种写几种），ajax 实现跨域的方法（客户端）

>- $.ajax(obje)

> ```

> $.ajax({

> type:'get' //'post'

> url:'/ajax/test'

> data:{usr:'w', p=12} //'usr=w&p=12'

> success: function(){} // x.readystate 4 && s.status 200

> error: function(){}

> })

> ```

>- $.get(url, data, function, dataTyep) //dataTyep 默认 text

> ```

> $.get('/ajax/test', {u='w'}, function(data){}, 'json')

> ```

>- $.post(obj)

> ```

> $.post('/ajax/test', {u='w'}, function(data){}, 'json')

> ```

>- $.getJSON(obj)

> ```

> $.getJSON('/ajax/test', {u='w'}, function(data){})

> ```

>- 跨域：$.getJSON(obj)

> ```

> $.getJSON('/ajax/test?callback=?', {u='w'}, function(data){})

> ```

82.编程:当用户点击 button 之后，将 ul 中的 li 加入一个 class 其值为“on”，并将 li 中的文本 a b c d e 依次 变为 ： 1 2 3 4 5

```
    <ul id="myUl">
        <li>a</li>
        <li>b</li>
        <li>c</li>
        <li>d</li>
        <li>e</li>
    </ul>
    <li>f<li>
    <input id="btn" type="button" value="button">
```

参考答案:

```
    $(function () {
        $('#btn').click(function () {
            $('#myUl>li').each(function () {
                var $this = $(this);
                var index = $(this).index();
                $this.addClass('on').html('' + (index + 1) );


            });
        })
```

});

额 这么说吧 服务器返回的是 json 字符串，但是你能看到的时候已经被前台转成对象了。

服务器当然可以传别的东西，但是你规定了按照 json 解析（其他的格式会解析出错）。

六、流行框架

1. 比较 var，let 和 const

* var : E7S6 之前用来定义变量的关键字，有变量提升，没有块作用域

* let : ES6 用来定义变量的关键字，没有变量提升，有块作用域

* const : ES6 用来定义常量的关键字,没有变量提升，有块作用域

2. 使用箭头函数编码设置定时器，不断输出当前时间

```
setInterval(() => console.log(Date.now()), 1000)
```

3. 编码演示对象表达增强和对象方法增强

```
*  对象表达增强
var name = 'tom';
function test() {...}
var obj = {name, test}
*  对象方法增强
{
    name : 'Tom',
    setName(name) {this.name = name}
}
```

4. 说说 promise 的作用，编码演示 promise 的基本使用

* 解决回调地狱(回调函数的层层嵌套，编码是不断向右扩展，阅读性很差)

```
var promise = new Promise(function(resolve, reject){
    //做异步的操作
    if(成功) {
        resolve(result);
    } else {
        reject(errorMsg);
    }
})
promise.then(function(
    result => console.log(result) ,
    errorMsg => alert(errorMsg)
)
```

5. 说说你对 Angular 双向数据绑定的理解

* View（视图）： 页面（标签、指令，表达式）
* Model(模型) ：作用域对象（属性、方法）
* 数据绑定： 数据从一个位置自动流向另一个位置
    * View-->Model
    * Model-->View
* 单向数据绑定： 只支持一个方向
    * View-->Model ： ng-init
    * Model-->View : {{name}}

* 双向数据绑定

    * Model<-->View : ng-model

6.说说你对 Angular 依赖注入的理解

* 依赖的对象被别人(调用者)自动注入进入

* 注入的方式;

    * 内部自建 ： 不动态

    * 全局变量 ： 污染全局环境

    * 形参： 这种最好

* angualr 就是通过声明式依赖注入， 来得到作用域对象

7.写出 10 个内置指令

* ng-app : 指定模块名， angular 管理的区域

* ng-model ： 双向绑定， 输入相关标签

* ng-init ： 初始化数据

* ng-click ： 调用作用域对象的方法（点击时）

* ng-controller : 指定控制器构造函数名， 内部会自动创建一个新的子作用域（外部的）

* ng-bind ： 解决使用{{}}显示数据闪屏（在很短时间内显示{{}}）

* ng-repeat ： 遍历数组显示数据， 数组有几个元素就会产生几个新的作用域 ='item in arr'

    * $index, $first, $last, $middle, $odd, $even

* ng-show : 布尔类型， 如果为 true 才显示

* ng-hide : 布尔类型， 如果为 true 就隐藏

* ng-class : 动态 class

* ng-style : 动态 style

* ng-mouseenter : 鼠标进入事件监听

* ng-focus : focus 事件监听

8.说说 scope 的$apply()与$watch()的作用

* $apply() : 强制进行脏数据检查, 用于非 angular 调用的回调函数中

* $watch() : 监视 scope 中某个属性数据的变化, 有一般监视和深度监视

9.说说你对 Angular 路由的理解和基本使用

* angular 中用来实现 SPA 的技术

* 在不进行页面跳转的情况下, 进行页面的局部更新显示(当请求某个路由路径时, 发送是 ajax 请求)

* 使用 angular-route.js

  * 将 angular-route.js 复制到当前项目中

  * 在页面中引入

  * 在创建 module 时依赖'ngRoute'

* 在主页中指定<ng-view>/<div ng-view>, 它是用来包含显示不同页面内容的容器

* 配置路由： 使用$routeProvider 服务

10.说说你对自定义指令中 scope 的理解

* scope ： false //直接使用外部作用域作为当前作用域

* scope ： true //新建一个作用域， 继承于外部作用域

* scope : { //隔离作用域： 新建一个作用域， 但不从外部外部作用域继承

  模板中需要显示： {{msg}}

  msg : '@' <my-directive msg='tt'> --->tt

  msg ： '=' <my-directive msg='tt/tt()'> ---->将 tt 作为表达式， 从外部作用域中取对应的属性显示

  模板中需要显示： {{msg({name:'Tom'})}}

msg : '&'　　<my-directive msg='tt(name)'>　---->将 tt(name)作为表达式，从外部作用域中取对应的方法调用来显示

}

11. 说说你对 js 模块和模块化编程的理解

* 什么是模块?

  * 将一个复杂的程序依据一定的规则(规范)封装成几个块(文件)，并进行组合在一起

  * 块的内部数据/实现是私有的，只是向外部暴露一些接口(方法)与外部其它模块通信

* 一个模块的组成

  * 数据--->内部的属性

  * 操作数据的行为--->内部的函数

* 模块化

  * 编码时是按照模块一个一个编码的，整个项目就是一个模块化的项目

11. 说说常用的模块化规范和实现

* CommonJS

  * Node.js

  * Browserify

* AMD

  * requirejs

* CMD

  * seajs

* ES6

  * ES6

12. 说说 ES6 的模块化

* 定义暴露模块 : export

  暴露一个对象:

  ```
  export default 对象
  ```

  暴露多个:

  ```
  export var xxx = value1
  export let yyy = value2
  ```

  ```
  var xxx = value1
  let yyy = value2
  export {xxx, yyy}
  ```

* 引入使用模块 : import

  default 模块:

  ```
  import xxx  from '模块路径/模块名'
  ```

  其它模块

  ```
  import {xxx, yyy} from '模块路径/模块名'
  import * as module1 from '模块路径/模块名'
  ```

* 解决 ES6 的模块化语法浏览器不能解析的问题?
  * 使用 Babel

13. 说说你在项目中是如何使用 requerjs 的

* 定义暴露模块:

  ```
  define([依赖模块名], function(){return 模块对象})
  ```

* 配置模块:

  ```
  require.config({
  ```

```
        //基本路径
        baseUrl : 'js/',
        //标识名称与路径的映射
        paths : {
            '模块 1' : 'modules/模块 1',
            '模块 2' : 'modules/模块 2',
            'angular' : 'libs/angular',
            'angular-messages' : 'libs/angular-messages'
        },
        //非 AMD 的模块
        shim : {
            'angular' : {
                exports : 'angular'
            },
            'angular-messages' : {
                exports : 'angular-messages',
                deps : ['angular']
            }
        }
    })
```

* 引入使用模块

```
require(['模块 1', '模块 2', '模块 3'], function(m1, m2){//使用模块对象})
```

14. 说说你对项目构建的理解和常用的项目构建工具

* 什么是项目构建?
    * 编译项目中的 js, sass, less
    * 合并 js/css 等资源文件

    * 压缩 js/css 等资源文件

* 构建工具的作用?

    * 简化项目构建，自动化完成构建

* 常用的项目构建工具

    * grunt

    * gulp

    * webpack

15. 说说你对 React 的理解

* Facebook 开源的一个 js 库，页向组件开发

* 它本身并不是一个 MVC 框架，而仅仅是其中的 V，用来动态渲染视图页面

* React 关心

    * 显示/更新 DOM

    * 响应事件

* React 高效的原因

    * 虚拟(virtual)DOM，不总是直接操作 DOM

    * 高效的 DOM Diff 算法，最小化页面重绘

16. 说说你对 JSX 的理解

* javascript XML

* 组成: js 的代码和 xml 标签

* 用来创建虚拟 DOM 对象的 JS 扩展语法，浏览器不能直接解析运行，需要 Babel 转译后才能运行

* js 中直接可以套 xml 标签，但 xml 标签要套 js 需要放在{}中

17. 编码实现一个简单的组件

* 创建组件类

```
//MyComp 是 constructor function
var MyComp = React.createClass({ //配置对象
    render : function(){ //自动添加到组件类的原型中去
        this instanceof MyComp --->true
        return (
            vDOM
        );
    }
})
```

* 渲染组件标签

    * ReactDOM.render(<MyComp />, cotainerEle);

18. 说说你对 vue 和 vue-router 的理解

* vue 是一位华裔前 Google 工程师开发的前端 js 库

* 与 angular.js 类似的是声明式开发，但性能高于 angular，体积小很多，比较适合移动端开发

* 它本身不是全能框架，只关注 UI，如果需要 router/ajax，可以使用对应插件或使用别的库来实现

* vue-router 是 vue 的路由扩展库(插件)，用于实现 SPA 应用

19. 创建 Vue 时选项对象中常用的几个属性

```
{
    el : '#app',
    data : {}/function
    methods : {}
    computed : {}
    watch : {}
```

```
        filters : {}
        redirectives : {}
        components : {}
    }
```

20. JQuery 的源码看过吗？能不能简单概况一下它的实现原理？

考察学习知识的态度，是否仅仅是停留在使用层面，要知其然知其所以然

21. jQuery.fn 的 init 方法返回的 this 指的是什么对象？为什么要返回 this？

this 执行 init 构造函数自身，其实就是 jQuery 实例对象，返回 this 是为了实现 jQuery 的链式操作

22. jquery 中如何将数组转化为 json 字符串，然后再转化回来？

```
$.parseJSON('{"name":"John"}');
```

23. jQuery 的属性拷贝(extend)的实现原理是什么，如何实现深拷贝？

递归赋值

24jquery.extend 与 jquery.fn.extend 的区别？

jquery.extend 用来扩展 jQuery 对象本身；jquery.fn.extend 用来扩展 jQuery 实例

25. JQuery 一个对象可以同时绑定多个事件，这是如何实现的？

可以同时绑定多个事件，低层实现原理是使用 addEventListner 与 attachEvent 兼容处理做事件注册

26.针对 jQuery 的优化方法？

优先使用 ID 选择器

在 class 前使用 tag(标签名)

给选择器一个上下文

慎用 .live()方法（应该说尽量不要使用）

使用 data()方法存储临时变量

七、移动端开发

1.定位：

position: absolute;默认的 left 和 top 的值是 auto,不是 0，会随文档流而变化

相对定位会参照他自己在文档流中的位置

绝对定位会参照定位父级，如果没有定位父级，会参照初始包含块

2.初始包含块

初始包含块：对于浮动元素，其包含块定义为最近的块级祖先元素，对于定位，CSS2.1 定义了以下行为：

1）“根元素”的包含块（也称为初始包含块）有用户代理(浏览器)建立。在 HTML 中，根元素就是 html 元素，

不过有些浏览器会使用 body 作为根元素，在大多数浏览器中，初始包含块是一个视窗大小的矩形。

2)对于一个非根元素，如果其 position 值是 relative 或 static,包含块则由最近的块级框、表单元格

或行内块祖先框的内容边界构成

3)对于一个非根元素，如果其 position 值是 absolute，包含块设置为最近的 position 值不是 static 的祖先元素，

过程如下：

-如果这个祖先是块元素，包含块则设置为该元素的内边距边界，换句话，就是由边框界定的区域

-如果没有祖先元素，元素的包含块定义为初始包含块

3. 布局

1)**三列布局**：两边固定，中间自适应，原来使用 flax 做(存在兼容性问题)

使用定位，给左右设置 position: absolute;left,top,

中间设置 margin 或 padding：0　左或右宽度，显示中间的内容，但是倾向于用 padding

使用 margin 的问题：外边距的问题（见 5）

padding：会渲染背景（如果两边渐变则不可以使用）:此时用 margin

body{min-width: 600px;}，最小宽度：2 倍的左边+右边，

但是，设置完最小宽度，浏览器的宽度小于 600，出现滚动条，存在问题

（内容区依旧被挤压，滑动滚动条，会在滚动条的后边会有内容区存在）

缺点：扩展性不好，会使绝对定位产生错位现象

内容区会被挤压，滑动滚动条，会在滚动条的后边会有内容区存在（因为左右依照适口大小而定位）

使用浮动：right 和 middle 的 div 调换位置，给左右设置浮动(浮动元素会提升半级)，

最好给中间也设置 padding：0　左或右宽度

body{min-width: 600px;}，最小宽度：2 倍的左边+右边，

不会出现上面的问题

缺点：扩展性不好，会使绝对定位产生错位现象

不符合浮动的本质，不推荐，推荐用 flex，但是兼容性不好

没有首先加载内容区

2)**圣杯布局**：浮动+margin 为负（核心）

1.两边固定,当中自适应

2.扩展性要好

3.极致的用户体验

实现：加头部加尾部-->为了扩展性好

内容区放在上边-->先加载内容区，为了用户体验需求

给左边设置：margin-left: -100% -->按文档流排列

给右边设置：margin-left: -200px（右边自己的宽度）

中间设置 padding：0 左或右宽度 --》有问题（宽度已经是 100%，在加 padding，宽度会发生变化），需要在 container 上加

需要在 container 上加：padding：0 左或右宽度 --》出现问题（两边会出现空白），需要给两边相对定位

两边相对定位：position:relative，分别设置 left 和 right

body 设置最小宽度：min-width

注意清除浮动：container 上，单独列一个属性.clearfix:after{content: "";clear: both;display: block;}

问题，left，right，middle 其中一个内容高度过高导致布局高度不统一，

一般给三者设置 min-height，但是不推荐，引出伪等高布局

伪等高布局解决上述问题：

容器内所有元素需要设置：left，right，middle

padding-bottom: 10000px;

margin-bottom: -10000px;

但是需要给容器设置 overflow：hidden

overflow：hidden :1,解决内容溢出 2，此属性也可以清除浮动

伪元素与伪类区别：伪类用来设置元素的状态 hover

伪元素 after，会创造出一个元素

缺点：浮动+margin 为负+定位 --》导致太复杂

没有实现等高布局

优点：html 结构清晰

留白使用 padding 来处理的：padding：0　左或右宽度　+

两边相对定位：position:relative，分别设置 left 和 right

内容溢出不会影响文档流

```
<style type="text/css">
    *{margin: 0;padding: 0;}
    body{min-width: 600px;}
    .container{padding: 0 200px;overflow: hidden;}
    .header,.footer{border: 1px solid;background: #aaa;text-align: center;}
    .middle,.left,.right{float: left;
            padding-bottom: 10000px;
            margin-bottom: -10000px;
            background: pink;}
    .left{position: relative;left: -200px;width: 200px;margin-left: -100%;}
    .right{position: relative;right:-200px ;width: 200px;margin-left: -200px;}
    .middle{width: 100%;background: deeppink;}
    .footer{clear: both;}

    .clearfix:after{content: "";clear: both;display: block;}
    </style>
    <div class="header">
        <h4>头部</h4>
    </div>
    <div class="container clearfix">
        <div class="middle">middle</div>
        <div class="left">
```

```
            left<br />
        </div>
        <div class="right">right</div>
    </div>
    <div class="footer">
        <h4>尾巴</h4>
    </div>
```

3)**伪等高布局**:

容器内所有元素需要设置：left，right，middle

padding-bottom: 10000px;

margin-bottom: -10000px;

但是需要给容器设置 overflow：hidden

overflow：hidden :1,解决内容溢出 2，此属性也可以清除浮动

4)**双飞翼布局**:浮动+margin 为负（核心）

双飞翼布局与圣杯布局的区别：处理留白不同，其他设置都一样（最小宽度，等高处理）

圣杯布局：使用 padding 来处理的：padding：0 200px +

两边相对定位：position:relative，分别设置 left 和 right

双飞翼布局：多加一层 div，用自身的 margin 解决

.middle-inner{margin: 0 200px;}

```
        <div class="middle">
            <div class="middle-inner">
                middle
            </div>
        </div>
```

4. overflow(溢出的部分不会对文档流造成影响)

6.1)visible　默认值。内容不会被修剪，会呈现在元素框之外。

hidden　内容会被修剪，并且其余内容是不可见的。

scroll　内容会被修剪，但是浏览器会显示滚动条以便查看其余的内容。

auto如果内容被修剪，则浏览器会显示滚动条以便查看其余的内容。

当父元素,子元素的高宽确定时,

overflow: auto;　父<子,出滚动条

overflow: scroll;　不管 父是不是<子,都出滚动条（只要设置 scroll，就会出现滚动条）

overflow: hidden; 子溢出的都会被截掉

6.1)不加 overflow:scroll;滚动条出现在了 html 的上一层 document 元素上

6.2)考虑 document html body:

1.html body 中只有一个有 overflow 属性时,这个属性会传给 document

2.当且仅当 html body 都存在 overflow 属性时.body 的 overflow 属性才能生效,

3.当 html body 都存在 overflow 属性时.body 的 overflow 属性一直都会作用于 body 身上,

html 的 overflow 属性一直都会作用于 document 身上

5. 禁止系统默认的滚动条

html{overflow:hidden;height: 100%;} -->高度与初始包含块保持统一

body{overflow:hidden;height: 100%;}

--》只作用在 body 自己身上，html 会作用在 document 上

6. jQuery 四大特征：

1）方法的链式调用

2）取赋值合体（读写二合一）

3）隐式迭代（可以给很多元素同时设置事件 click，jQuery 内部会做出很多我们看不见的工作）

4）函数化编程（封装成函数来操作，核心函数$和核心对象$()）

7.attr&prop 的区别，以及在项目中的运用

定义：

attribute:html 标签中的属性(预定义属性,自定义属性)

property：浏览器帮我们解析出来的 dom 对象的属性

原生对象：我们自己定义的属性和__proto__

dom 对象：由浏览器帮我们解析出来的，

里面包含了 attributies 属性：html 标签里面的预定义属性和自定义的属性的集合

attributies 是 property 其中的一个属性(但并不是它的子集)

attributies 里面的对象叫属性节点

attributies 属性中，html 标签里面的预定义属性，在 property 的属性单独存在（自定义属性不会存在）

每一个 attribute 都有与之对应的 property

区别：

非布尔值属性:

attribute 和 property 会实时同步

布尔值属性:

如果修改了 property,与之对应的 attribute 不会同步

如果修改了 attribute:

没有操作过 property,attribute 会实时同步 property

一旦操作了 property,attribute 不会实时同步 property

浏览器的读写操作,只会关注 property 属性

(那些我们可以通过浏览器端通过实际操作来改变的属性最好使用 prop(),比如 input)

特殊：property 的 id 对应 attribute 的"id"，

property 的 className 对应 attribute 的"class"(class 是保留字)

8. 清除浮动：让浮动的子元素把父元素撑开

解决问题：

1.给父元素设置高度（扩展性不好）

开启了 BFC

2、给父级加浮动 (页面中所有元素都加浮动，margin 左右自动失效)

3、设置 overflow

clear 的特性

4、空标签清除浮动 (IE6 最小高度 19px；（解决后 IE6 下还有 2px 偏差）)

兼容性:IE6 最小高度 19px,即一切高度小于 19px 的元素在 IE6 下都会被渲染成 19px

在浮动元素后边添加：

1）.clear{clear: both;} --》不设置高度可以正常显示

2）.clear{height: 0;clear: both;}

设置高度时，IE6 会有 19px 的空白高度，此时在父级上设置 font-size：0 --》IE6 下还有 2px 空白行

缺点： 1，多添加标签，破坏结构样式。

2，此空白标签不能添加内容，不利于后期的维护

3，添加高度，IE6 兼容问题。

5、br 清浮动(不符合工作中：结构、样式、行为，三者分离的要求)

在浮动元素后边添加：<br clear='all'/>

br 中的 clear 属性(attributes)对应的不是 css 中的 clear(proterty)

clear 的特性 + 触发了 haslayout

6、伪元素 after(不支持 IE6、IE7)

:after{content: "";display: block;clear: both;}

IE6、IE7 给父级添加{*zoom：1}　　--》*，只在 IE6、IE7 中识别，提高性能

清除浮动模板：

```
<style>
    .clearfix{
        *zoom:1;
    }
    .clearfix:after{
        content: "";
        display: block;
        clear: both;
    }
    .box{
        border:1px solid red;
    }
    .item{
        width: 200px;
        height: 200px;
        background-color: red;
        float: left;
    }
```

```
</style>


<div class="box clearfix">
    <div class="item"></div>
</div>
```

9. BFC（块级格式化上下文）

BFC 布局规则：

1.内部的 Box 会在垂直方向，一个接一个地放置。

2.内部的 Box 垂直方向的距离由 margin 决定。属于同一个 BFC 的两个相邻 Box 的 margin 会发生重叠

3.每个元素的 margin box 的左边， 与包含块 border box 的左边相接触(对于从左往右的格式化，否则相反)。即使存在浮动也是如此。

4.BFC 的区域不会与 float box 重叠。

5.BFC 就是页面上的一个隔离的独立容器，容器里面的子元素不会影响到外面的元素。反之也如此。

6.计算 BFC 的高度时，浮动元素也参与计算。

BFC 什么时候出现(哪些元素会生成 BFC?)

根元素

float 属性不为 none

position 为 absolute 或 fixed

overflow 不为 visible

display 为 inline-block, table-cell, table-caption, flex, inline-flex

10. 元素垂直水平居中

1）绝对定位盒模型的特点:

定位盒子的盒模型大小 + 偏移量 = 包含块的尺寸

缺点：需要通过操作宽度和高度，指定宽度和高度

案例一：

```
position: absolute;
left: 50%;  //百分比参照父元素
top: 50%;
margin-left:-100px ;
margin-top:-100px ;
```

案例二：

```
position: absolute;
left: 0;
right: 0;
top: 0;
bottom: 0;
margin:auto;
```

.2）3d 变换

优点：不需要通过操作宽度和高度

```
position: absolute;
left: 50%;
top: 50%;
-webkit-transform: translate3d(-50%, -50%, 0);//百分比参照自身
transform: translate3d(-50%, -50%, 0);
```

11. 首尾去空格

技术点：正则的匹配 var reg = /^\s+|\s+$/g

```
<script type="text/javascript">
    var str = '      hello      '

    String.prototype.trim = function(){
        //var reg = /^\s+|\s+$/g      //($hello$)
        var reg = /^\s*|\s*$/g    //($hello$$)

        return this.replace(reg,'$')
    }
    console.log('('+str.trim()+')')
</script>
```

区别：+ 匹配至少一个 x

* 匹配 0 个或任意多个 x

12.自定义 match 方法

```
console.log(match(/f/g,"fffff")); //["f", "f", "f", "f", "f"]

function match(reg,str){
    var result = reg.exec(str);
    //console.log(result)    -->["f", index: 0, input: "fffff"]
    var arr=[];
    if(reg.global){
        while(result){
            arr.push(result[0]);
            result = reg.exec(str);//数组更新
            //console.log(result)
        }
```

```
        }else{
            arr.push(result[0])
        }
        return arr;
    }
```

13. 面试：寻找重复最多的字符以及个数

```
<script type="text/javascript">
    var str = 'shdshdfjkfjfdgjkjdksgjskdjfsfsfsfjksjkfdkjf'
    var arr = str.split('').sort()//把字符串变为数组，在排序
    str = arr.join('') //把排序好的数组，变为字符串

    var count = 0   //不能使用 undefined，转化为数字是 NaN
    var char = 0

    var reg = /(\w)\1+/g
    str.replace(reg,function(parent,son){
        if(parent.length >count){
            count = parent.length
            char = son
        }
    })
    console.log("最多的字符为:"+char+";个数为:"+count)
</script>
```

14. 点击 a 标签，删除对应的父节点 li

```
<ul>
    <li>nmx1<a href="javascript:;">删除</a></li>
    <li>nmx2<a href="javascript:;">删除</a></li>
    <li>nmx3<a href="javascript:;">删除</a></li>
</ul>
```

方法一：使用 let 解决(let 有兼容性,ES6)

```
var a = document.getElementsByTagName('a')
for (let i= 0;i<a.length;i++) {
    a[i].onclick = function (){
        a[i].parentNode.style.display = 'none' //不会影响 dom 结构，节约浏览器开销
    }
}
```

方法二：使用 this 解决

```
var a = document.getElementsByTagName('a')
for (var i= 0;i<a.length;i++) {
    a[i].onclick = function(){
        this.parentNode.style.display = 'none'
    }
}
```

方法三：立即执行函数表达式 IIFE，闭包原理

```
var a = document.getElementsByTagName('a')
for (var i= 0;i<a.length;i++) {
    (function(j){
        a[j].onclick = function(){
            a[j].parentNode.style.display='none'
        }
```

```
        })(i)
    }
```

15. 获取元素的绝对坐标(自定义)

```
<script type="text/javascript">
var div5 = document.getElementById('div5')
console.log(getPos(div5))

function getPos(node){
    //设置一个初始位置
    var pos={left:0,top:0};
    //对每一个节点进行遍历
    while(node){
        pos.left += node.offsetLeft; //累加节点
        pos.top += node.offsetTop;
        node = node.offsetParent   //每循环一次，向上传一个父节点
    }
    return pos
}
```

16. 获取元素的相对坐标

```
<script type="text/javascript">
    var div5 = document.getElementById('div5')
    window.onscroll = function(){
        console.log(getPos(div5))
    }
```

```
        function getPos(node){
            var left = document.documentElement.scrollLeft ||
document.body.scrollLeft;
            var top = document.documentElement.scrollTop ||
document.body.scrollTop;
            //设置初始值
            var scroll = {left:left,top:top}
            var pos = {left:0,top:0}

            while(node){    //匹配到最后返回 null
                pos.left += node.offsetLeft
                pos.top += node.offsetTop
                node = node.offsetParent
            }
            return {left:pos.left-scroll.left , top:pos.top-scroll.top}
        }
    </script>
```

17.如何检测 ie 版本

```
    <script type="text/javascript">
        console.log(isIe(9));

        function isIe(version){
            var b = document.createElement("b");
            b.innerHTML="<!--[if lte IE "+version+"]><i></i><![endif]-->";
```

```
            var result =b.getElementsByTagName("i").length == 1;
            return result;
        }
    </script>
```

18. 滚轮事件（处理兼容代码）

```
var testNode = document.getElementById("test");
    testNode.onmousewheel=fn;
    test.addEventListener && test.addEventListener('DOMMouseScroll' ,fun)

    function fn(ev){
        var e = ev||event;

        //方向
        var flag="";
        if(e.wheelDelta){
            //正:向上,负:向下
            flag = e.wheelDelta>0?"up":"down";
        }else if(e.detail){
            //正:向下,负:向上
            flag = e.detail>0?"down":"up";
        }

        switch (flag){
            case "up":
                testNode.style.height = testNode.offsetHeight -10+"px";
                break;
            case "down":
```

```
                testNode.style.height = testNode.offsetHeight +10+"px";
                break;
            }

            test.addEventListener && event.preventDefault()
            return false;
        }
```

19. 区分 onmouseover, onmouseout, onmouseenter, onmouseleave

onmouseover,onmouseout 会发生冒泡事件，会传递到上一级

onmouseenter,onmouseleave 单纯的划入划出，里边的元素不会受到影响

20. 视口

document.documentElement.clientHeight --》（最干净的那个视口）的高度（没有兼容性问题）

window.innerHeight --》视口（包括滚动条的尺寸）（几乎所有浏览器全部支持）

window.outerHeight --》浏览器窗口，包括外部的高度，包括工具栏，以及浏览器外边的阴影部分

screen.width --》视口（包括滚动条的尺寸） 它在我们的移动端返回的不一定是我们屏幕的分辨率，它可能返回我们屏幕尺寸，不推荐使用（有很大的兼容问题）

21. PC 端事件在移动端会有 300ms 延迟的问题

click:点击事件，300ms 后遮罩消失，只触发 click 事件，但是，click 在移动端会默认触发 touchstart 事件，300ms 后，click 事件让遮罩消失

touchstart：点击事件，首先立即响应，遮罩消失，然后 300ms 后，会默认触发 click 事件，页面的 a 标签发生跳转，此时 click 作用在 a 标签上

所以，基于上述问题：需要取消系统的事件默认行为

```
document.addEventListener("touchstart",function(ev){
    ev.preventDefault();
})
```

22. 事件绑定通用代码

```
function bind(obj, evname, fn) {
    if (obj.addEventListener) {//除 ie 低版本外都可以进入
        obj.addEventListener(evname, fn, false);
    } else {
        obj.attachEvent('on' + evname, function() {
            fn.call(obj);
        });
    }
}

bind(document, 'click', fn1);
```

23. em 与 rem

em:基于自身的 font-size（如果自身没有设置 font-size,它会继承父元素 font-size）

rem:rem 基于 html 的 font-size

24. 适配：等比缩放

rem 适配方案：通过 js 动态设置 html 字体的大小

IIFE 避免污染全局

```
(function(){
    var html = document.querySelector("html");
    var width = document.documentElement.clientWidth;
    //设置字体转化为 rem，此时求出的是 1rem
    html.style.fontSize =width/16 +"px"; //16 自己定义的，一般习惯使用 16 或者 10
})()
```

25. viewport 适配

给的设计图本身就是 320（使布局适口变为 320）

```
<script type="text/javascript">
    window.onload = function(){
        var target = 320
        var scalce = screen.width/target
        //创建 meta 标签
        var meta = document.createElement('meta')
        meta.setAttribute('name','viewport')
        meta.setAttribute('content','initial-scale='+scalce)
        document.head.appendChild(meta)
    }
</script>
```

viewport 适配优点：特别简单，无需做 rem 的转化

给的设计图 750

```
<meta name="viewport" content="width=device-width"/>
```

```
<script type="text/javascript">
    window.onload = function(){
        var target = 750;
        var scale = document.documentElement.clientWidth/750;
        var meta =document.querySelector("meta[name='viewport']");
        //覆盖操作
        meta.setAttribute("content","initial-scale="+scale );

        //设置字体
        var body = document.querySelector("body");
        body.style.fontSize=16/scale+"px";
    }
</script>
```

八、NodeJs

1.如何判断当前脚本运行在浏览器还是 node 环境中？（阿里）

```
this === window ? 'browser' : 'node';
```

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

2.对 Node 的优点和缺点提出了自己的看法？

（优点）因为 Node 是基于事件驱动和无阻塞的，所以非常适合处理并发请求，

因此构建在 Node 上的代理服务器相比其他技术实现（如 Ruby）的服务器表现要好得多。

此外，与 Node 代理服务器交互的客户端代码是由 javascript 语言编写的，

因此客户端和服务器端都用同一种语言编写，这是非常美妙的事情。

（缺点）Node 是一个相对新的开源项目，所以不太稳定，它总是一直在变，

而且缺少足够多的第三方库支持。看起来，就像是 Ruby/Rails 当年的样子。

3.需求：实现一个页面操作不会整页刷新的网站，并且能在浏览器前进、后退时正确响应。给出你的技术实现方案？

至少给出自己的思路（url-hash,可以使用已有的一些框架 history.js 等）

4.Node.js 的适用场景？

1)、实时应用：如在线聊天，实时通知推送等等（如 socket.io）

2)、分布式应用：通过高效的并行 I/O 使用已有的数据

3)、工具类应用：海量的工具，小到前端压缩部署（如 grunt），大到桌面图形界面应用程序

4)、游戏类应用：游戏领域对实时和并发有很高的要求（如网易的 pomelo 框架）

5)、利用稳定接口提升 Web 渲染能力

6)、前后端编程语言环境统一：前端开发人员可以非常快速地切入到服务器端的开发（如著名的纯 Javascript 全栈式 MEAN 架构）

5.(如果会用 node)知道 route, middleware, cluster, nodemon, pm2, server-side rendering 么?

Nodejs 相关概念的理解程度

6.什么是“前端路由”?什么时候适合使用“前端路由”？“前端路由”有哪些优点和缺点?

熟悉前后端通信相关知识

7. 对 Node 的优点和缺点提出了自己的看法？

优点：

1. 因为 Node 是基于事件驱动和无阻塞的，所以非常适合处理并发请求，因此构建在 Node 上的代理服务器相比其他技术实现（如 Ruby）的服务器表现要好得多。

2. 与 Node 代理服务器交互的客户端代码是由 javascript 语言编写的，因此客户端和服务器端都用同一种语言编写，这是非常美妙的事情。

缺点：

1. Node 是一个相对新的开源项目，所以不太稳定，它总是一直在变。

2. 缺少足够多的第三方库支持。看起来，就像是 Ruby/Rails 当年的样子（第三方库现在已经很丰富了，所以这个缺点可以说不存在了）。

九、前端概括性问题

1. 你有用过哪些前端性能优化的方法？

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

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

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

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

（5） 少用全局变量、缓存 DOM 节点查找的结果。减少 IO 读取操作。

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

（7） 图片预加载，将样式表放在顶部，将脚本放在底部 加上时间戳。

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

对普通的网站有一个统一的思路，就是尽量向前端优化、减少数据库操作、减少磁盘 IO。

向前端优化指的是，在不影响功能和体验的情况下，能在浏览器执行的不要在服务端执行，能在缓存服务器上直接返回的不要到应用服务器，程序能直接取得的结果不要到外部取得，本机内能取得的数据不要到远程取，内存能取到的不要到磁盘取，缓存中有的不要去数据库查询。

减少数据库操作指减少更新次数、缓存结果减少查询次数、将数据库执行的操作尽可能的让你的程序完成（例如 join 查询），减少磁盘 IO 指尽量不使用文件系统作为缓存、减少读写文件次数等。程序优化永远要优化慢的部分，换语言是无法“优化”的。

2. 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。

1、实现界面交互

2、提升用户体验

3、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，

参与项目，快速高质量完成实现效果图，精确到 1px；

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；

处理 hack，兼容、写出优美的代码格式；

针对服务器的优化、拥抱最新前端技术。

3. 最近在看哪些前端方面的书？

javascript 权威指南， javascript 高级程序设计，ES6 入门，Nodejs

4. 平时如何管理你的项目，如何设计突发大规模并发架构？

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

编写习惯必须一致（例如都是采用继承式的写法，单样式都写成一行）；

标注样式编写人，各模块都及时标注（标注关键样式调用的地方）；

页面进行标注（例如 页面 模块 开始和结束）；

CSS 跟 HTML 分文件夹并行存放，命名都得统一（例如 style.css）

JS 分文件夹存放 命民以该 JS 功能为准英文翻译；

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

5. 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）

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

Node.js、Mongodb、npm、react、angularjs、MVVM、MEAN

7. 你有了解我们公司吗？说说你的认识？

因为我想去阿里，所以我针对阿里的说

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

7. 移动端（比如：Android IOS）怎么做好用户体验?

融入自己的设计理念，注重用户体验，选择合适的技术

8. 你所知道的页面性能优化方法有那些？

压缩、合并，减少请求，代码层析优化。。。

9. 除了前端以外还了解什么其它技术么？你最最厉害的技能是什么？

知识面宽度，最好熟悉一些后台语言，比如 php，展现出自己的技术两点

10. 谈谈你认为怎样做能使项目做的更好？

考虑问题的深入，不仅仅停留在完成任务上，要精益求精

11. 你对前端界面工程师这个职位是怎么样理解的？它的前景会怎么样？

表现出对前端的认同与兴趣，关注相关技术前沿

12. 如何优化网页加载速度？

1.减少 css，js 文件数量及大小(减少重复性代码，代码重复利用)，压缩 CSS 和 Js 代码

2.图片的大小

3.把 css 样式表放置顶部，把 js 放置页面底部

4.减少 http 请求数

5.使用外部 Js 和 CSS

13. 对前端界面工程师这个职位是怎么样理解的？它的前景会怎么样？

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。

1、实现界面交互

2、提升用户体验

3、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，

参与项目，快速高质量完成实现效果图，精确到 1px；

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；

处理 hack，兼容、写出优美的代码格式；

针对服务器的优化、拥抱最新前端技术。

其它相关的加分项：

1. 都使用和了解过哪些编辑器?都使用和了解过哪些日常工具?

2. 都知道有哪些浏览器内核?开发过的项目都兼容哪些浏览器?

3. 瀑布流布局或者流式布局是否有了解

4. HTML5 都有哪些新的 API?

5. 都用过什么代码调试工具?

6. 是否有接触过或者了解过重构。

7.你遇到过比较难的技术问题是？你是如何解决的？