自我介绍

一、HTML

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、HTML5 为什么只需要写 <!DOCTYPE HTML>?

HTML5 不基于 SGML（通用标记语言），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。

3、行内元素有哪些？块级元素有哪些？ 空(void)元素有那些?

（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~h6 p

（3）常见的空元素：

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

4、页面导入样式时，使用 link 和@import 有什么区

别？

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;

5、对浏览器内核的理解？

Trident 内核：IE,搜狗浏览器等。[又称 MSHTML]
Gecko 内核：FF 等
Presto 内核：Opera7 及以上。
Webkit 内核：Safari,Chrome 等。

6、对引擎的理解？

主要分成两部分：渲染引擎(layout engineer 或 Rendering Engine)和 JS 引擎。

渲染引擎：负责取得网页的内容（HTML、XML、图像等等）、整理讯息（例如加入 CSS 等），以及计算网页的显示方式，然后会输出至显示器或打印机。浏览器的内核的不同对于网页的语法解释会有不同，所以渲染的效果也不相同。所有网页浏览器、电子邮件客户端以及其它需要编辑、显示网络内容的应用程序都需要内核。

JS 引擎则：解析和执行 javascript 来实现网页的动态效果。

最开始渲染引擎和 JS 引擎并没有区分的很明确，后来 JS 引擎越来越独立，内核就倾向于只指渲染引擎。
欢迎加入全栈开发交流群一起学习交流：864305860

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

* HTML5 现在已经不是 SGML 的子集，主要是关于图像，位置，存储，多任务等功能的增加。

绘画 canvas;
用于媒介回放的 video 和 audio 元素;

本地离线存储 localStorage 长期存储数据，浏览器关闭后数据不丢失;
sessionStorage 的数据在浏览器关闭后自动删除;
语意化更好的内容元素，比如 article、footer、header、nav、section;
表单控件，calendar、date、time、email、url、search;
新的技术 webworker, websocket, Geolocation;

移除的元素：
纯表现的元素：basefont，big，center，font, s，strike，tt，u;
对可用性产生负面影响的元素：frame，frameset，noframes；

* 低版本 IE 支持 HTML5 新标签：

IE8/IE7/IE6 支持通过 document.createElement 方法产生的标签，
可以利用这一特性让这些浏览器支持 HTML5 新标签，
浏览器支持新标签后，还需要添加标签默认的样式。

当然也可以直接使用成熟的框架、比如 html5shim;

```
<!--[if lt IE 9]>
    <script> src="http://html5shim.googlecode.com/svn/trunk/html5.js"</script>
<![endif]-->
```

* 如何区分 HTML5： DOCTYPE 声明\新增的结构元素\功能元素

欢迎加入全栈开发交流群一起学习交流：864305860

8、简述一下你对 HTML 语义化的理解？

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

9、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。

10、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），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

11、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**。

12、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>
```

13、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>
```

14、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。

15、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；

二、CSS

1、css 盒子模型

内容(content)、填充(padding)、边框(border)、边界(margin)， CSS 盒子模式都具备这些属性

2、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
* 不可继承的样式：border padding margin width height ;

3、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

4、CSS3 新增伪类有那些？

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

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

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

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

p:nth-child(2) 选择属于其父元素的第二个子元素的每个 <p> 元素。

::after 在元素之前添加内容,也可以用来做清除浮动。

::before 在元素之后添加内容

:enabled

:disabled 控制表单控件的禁用状态。

:checked 单选框或复选框被选中。

5、如何居中 div？

1、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 */
}
```

2、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: absolute;        /* 绝对定位 */
    width:500px;
    height:300px;
    top: 50%;
    left: 50%;
    margin: -150px 0 0 -250px;          /* 上、左外边距为自身宽高的一半的负值 */
    background-color: pink;        /* 方便看效果 */

 }
```

3、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%);     /*位移*/
    background-color: pink;        /* 方便看效果 */
```

```
}
```

4、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 */
    height: 500px;              /*方便看效果*/
    border:1px solid red;

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

6、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。

7、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（默认相对 body 进行定位）。
fixed （老 IE 不支持）
生成固定定位的元素，相对于浏览器窗口进行定位。

relative
生成相对定位的元素，相对于其它初始位置进行定位。
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。

8、CSS3 有哪些新特性？

新增各种 CSS 选择器 （: not(.input)：所有 class 不是“input”的节点）
圆角 （border-radius:8px）
多列布局 （multi-column layout）
阴影和反射 （Shadow\Reflect）
文字特效 （text-shadow、）
文字渲染 （Text-decoration）
线性渐变 （gradient）
旋转 （transform）
缩放,定位,倾斜,动画,多背景
例如:transform:\scale(0.85,0.90)\ translate(0px,-30px)\ skew(-9deg,0deg)\Animation:

9、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。
适用场景：
Flexbox 适用于包含有多个元素的盒子的样式渲染
Flexbox 适用于在子元素的尺寸未知或者动态的情况下，对子元素的对齐方式、排列方式以及排序顺序进行控制展示

10、纯 CSS 创建一个三角形的原理是什么？

把上、左、右三条边框隐藏掉（颜色设为 transparent）

```
#demo {
    width: 0;
    height: 0;
    border-width: 20px;
    border-style: solid;
    border-color: transparent transparent red transparent;
}
```

11、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 使其不换行即可

12、css 多列等高如何实现？

1、利用 padding-bottom|margin-bottom 正负值相抵（子元素设置）；

2、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，

3、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，

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

```
<div id="container">
    <div class="column">
        <p>Sidebar</p>
    </div>
    <div class="column">
        <p>Main content；
            content；
            <br>
            <br>
            content；content；
            content</p>
    </div>
    <div class="column">
        <p>Sidebar</p>
    </div>
</div>
```

```
#container {
   margin: 0 auto;
   overflow: hidden;
   width: 960px;
}

.column {
   background: #ccc;
   float: left;
   width: 200px;
   margin-right: 5px;
   margin-bottom: -99999px;
   padding-bottom: 99999px;
}
```

13、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 ？

1、浏览器默认的 margin 和 padding 不同。

解决方案是给文档中用到的标签设置：margin:0;padding:0;。

2、低版本 ie 浏览器下 css 样式问题（给低版本 ie 浏览器识别的特定 css 样式）

```
.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*/
 }
```

3、 IE 下,可以使用获取常规属性的方法来获取自定义属性（**元素 . 自定义属性名**）,

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**。

4、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。

5、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 {}

14、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。

15、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。

16、BFC 的布局规则以及触发条件

Block Formatting Contexts（BFC）：**块级元素格式化上下文**

它决定了块级元素如何对它的内容进行布局，以及与其他元素的关系和相互关系

块级元素：父级（是一个块元素）

内容：子元素（是一个块元素）

其他元素：与内容同级别的兄弟元素

相互作用：BFC 里的元素与外面的元素不会发生影响

触发（创建）BFC 的方式（一下任意一条就可以）：

1.float 的值不为 none

2.overflow 的值不为 visible

3.display 的值为 table-cell、tabble-caption 和 inline-block 之一

4.position 的值不为 static 或者 releative 中的任何一个

2．BFC的特点

W3C 标准描述 BFC 的特点共有两条：

（1）在一个 BFC 中，盒子从顶端开始垂直一个接着一个地排列，两个相邻盒子之间的垂直间距由 margin 属性决定。在同一个 BFC 中，两个相邻块盒子之间垂直方向上的外边距会叠加。

（2）在一个 BFC 中，每一个盒子的左外边界（margin-left）会紧贴着容器的左边（border-left）（对于从右到左的格式化，则相反），即使存在浮动元素也是如此。

从上面的 W3C 标准定义，我们可以得出以下几点重要结论（非常重要，请字斟句酌地理解记忆）。

（1）在一个 BFC 内部，盒子会在垂直方向上一个接着一个地排列。

（2）在一个 BFC 内部，相邻的 margin-top 和 margin-bottom 会叠加。

（3）在一个 BFC 内部，每一个元素的左外边界会紧贴着包含盒子的左边，即使存在浮动也是如此。

（4）在一个 BFC 内部，如果存在内部元素是一个新的 BFC，并且存在内部元素是浮动元素。则该 BFC 的区域不会与 float 元素的区域重叠。

（5）BFC 就是页面上的一个隔离的盒子，该盒子内部的子元素不会影响到外面的元素。

（6）计算一个 BFC 的高度时，其内部浮动元素的高度也会参与计算。

17、css 定义的权重

权重的规则：标签的权重为 1，class 的权重为 10，id 的权重为 100，等等；
如果权重相同，则最后定义的样式会起作用。

18、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。有时候浮动没处理好也会导致文档错乱情况。

1、父级 div 定义 height；

2、父级 div 也一起浮动；

3、浮动元素的父级 div 定义伪元素::after；

```
.clearfix::after {
    content: " ";
    display : block;
    clear:both;
    overflow:hidden;
    *zoom: 1;
}
```

19、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。
1、当一个 div 在另一个 div 里，两个 div 都有 margin 的属性，想让这两个 div 不出现外边距合并的问题，必须给外层 div 添加 border 属性或者 overflow:hidden 属性；如果允许其出现外边距合并，内层 div 始终保持原位置，而外层 div 的 margin 属性会取两者的最大值。

2、当两个 div 不存在嵌套关系时（而是上下关系）一个设置了 margin-bottom，一个设置了 margin-top，不管是否有 border 或者 overflow 属性时，两者始终会存在外边距合并问题。

20、CSS 预处理器

Sass 、less 这些
Sass 语法是$
less 语法是@

21、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）；
尽量不用后代选择器，少用子代选择器，最好不要超过三层；
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）；

22、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）元素之间有一段空间

23、什么是响应式设计？响应式设计的基本原理是什

么？

响应式网页设计就是一个网站能够兼容多个终端——而不是为每个终端做一个特定的版本。这样，我们就可以不必为不断到来的新设备做专门的版本设计和开发了。

响应式设计的基本原理是通过媒体查询检测不同的设备屏幕尺寸做处理。页面头部必须有 meta 声明 viewport：

<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no”>

24、::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。

25、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);
}
```

26、box-sizing 属性

box-sizing 属性主要用来控制元素的盒模型的解析模式。默认值是 content-box。

content-box：让元素维持 W3C 的标准盒模型。元素的宽度/高度由 border + padding + content 的宽度/高度决定，设置 width/height 属性指的是 content 部分的宽/高

border-box：让元素维持 IE 传统盒模型（IE6 以下版本和 IE6~7 的怪异模式）。设置 width/height 属性指的是 border + padding + content

box-sizing

box-sizing属性可以为三个值之一：content-box (default)，border-box，padding-box。

content-box，border和padding不计算入width之内

padding-box，padding计算入width内

border-box，border和padding计算入width之内，其实就是怪异模式了~

ie8+浏览器支持content-box和border-box;

ff则支持全部三个值。

使用时：

-webkit-box-sizing: 100px; // for ios-safari, android

-moz-box-sizing:100px; //for ff

box-sizing:100px; //for other

27、Css 实现多行文字的垂直居中

1、法一　模拟单行文本来实现多行文本居中

外层 div 设置高和行高相同，控制一行能垂直居中，然后多行里面的 span 标签设置私有的行高，让多行文本能正常显示。（缺点，多行文本不能超过父容器的高度，不然垂直居中效果消失）

```
<style type="text/css">
        div{float: left;width: 200px;height:200px;line-height: 200px;margin: 10px;border:1px
            solid blue; }
        span{display: inline-block;vertical-align: middle;line-height: 22px;}
</style>

<div>
    <span>测试文字测试文字</span>
```

```
</div>
<div>
    <span>
    测试文字 <br/> 测试文字<br/> 测试文字<br/> 测试文字<br/> 测试文字<br/> 测试文字
    </span>
</div>
```

2、法二　display:table-cell 属性来模拟表格

等高布局，无需设置高度（也可以设置），文字轻松实现垂直居中

```
<style type="text/css">
        div{display: table-cell;width: 200px;height: 200px;border:1px solid blue;vertical-align: middle;}
</style>


<div>
    <span>测试文字测试文字</span>
</div>
<div>
    测试文字测试文字　<br/> 测试文字
</div>
<div>
    <span>测试文字 <br/> 测试文字<br/> 测试文字<br/> 测试文字<br/> 测试文字<br/> 测试文字</span>
</div>
```

3、法三　padding 法

```
<style type="text/css">
        .div4{
            border:1px solid red;
            width:400px;
            padding:30px 0;
            text-align:left;
```

```
        }
</style>

<div class="div4">
    div 中多行文字垂直水平居中
    <br>
    div 中多行文字垂直水平居中
    <br>
    div 中多行文字垂直水平居
</div>
```

28、link 和@import 引入 CSS 的区别？

区别 1：link 是 XHTML 标签，除了加载 CSS 外，还可以加载其他，如<link href="ico/ico.ico" rel="Shortcut Icon" />

；@import 属于 CSS 范畴，只能加载 CSS。

区别 2：link 引用 CSS 时，在页面载入时同时加载；@import 需要页面网页完全载入以后加载。

区别 3：link 是 XHTML 标签，无兼容问题；@import 是在 CSS2.1 提出的，低版本的浏览器不支持。

区别 4：link 支持使用 Javascript 控制 DOM 去改变样式；而@import 不支持。

三、JS

1、js 如何判断一个数组

4.通用的方法

```
var ary = [1,23,4];
function isArray(o){
return Object.prototype.toString.call(o)=='[object Array]';
}
console.log(isArray(ary));
```

是的话返回 true

2、ajax 的优缺点

ajax 的优点

1、最大的一点是页面无刷新，在页面内与服务器通信，给用户的体验非常好。

2、使用异步方式与服务器通信，不需要打断用户的操作，具有更加迅速的响应能力。

ajax 的缺点

1、ajax 干掉了 back 按钮，即**对浏览器后退机制的破坏**。后退按钮是一个标准的 web 站点的重要功能，但是它没法和 js 进行很好的合作。用户往往是希望能够通过后退来取消前一次操作的。那么对于这个问题有没有办法？答案是肯定的，用过 Gmail 的知道，Gmail 下面采用的 ajax 技术解决了这个问题，在 Gmail 下面是可以后退的，但是，它也并不能改变 ajax 的机制，它只是采用的一个比较笨但是有效的办法，即**用户单击后退按钮访问历史记录时，通过创建或使用一个隐藏的 IFRAME 来重现页面上的变更。**（例如，当用户在 Google Maps 中单击后退时，它在一个隐藏的 IFRAME 中进行搜索，然后将搜索结果反映到 Ajax 元素上，以便将应用程序状态恢复到当时的状态。）

3、js 中严格模式

1. 不允许用 with。

2. 所有变量必须声明，赋值给未声明的变量报错，而不是隐匿创建全局变量。

3. eval 中的代码不能创建 eval 所在作用域下的变量、函数。而是为 eval 单独创建一个作用域，并在 eval 返回时丢弃。

4. 函数中的特殊对象 arguments 是静态副本，而不像非严格模式那样，修改 arguments 或修改参数变量会相互影响。

5. 删除 configurable=false 的属性时报错，而不是忽略。

6. 对象字面量重复属性名报错。

7. 禁止八进制字面量，如 010（八进制的 8）。

8. 严格模式下 eval、arguments 变为关键字，不能用作变量名。

9. 一般函数调用时（不是对象的方法调用，也不使用 apply/call/bind 等修改 this）this 指向 null，而不是全局变量。

10. 试图修改不可写属性（writable=false），在不可扩展的对象上添加属性时报 TypeError，而不是忽略。

11. arguments.caller,arguements.callee 被禁用

4、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)

5、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

6、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，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。

7、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。

8、如何将字符串转化为数字，例如’12.3b’？

```
function toNum( str ){
    var reg = /^\d{1,}\.*\d*/;        //正则表达式匹配
    var strNum = str.match(reg);
    if( strNum !== null ){
        strNum = parseFloat(strNum );
    }
    return strNum ;
}
```

9、实现数组的随机排序？

```
var arr = [1,2,3,4,5,6,7,8,9,10];
    arr.sort(function(){
        return Math.random() - 0.5;
    })
    console.log(arr);
```

10、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。

11、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；

12、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 +')');

13、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，是 window 对象的一个属性。

14、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　表示一个变量用 var 声明了但有初始化(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；

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
```

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、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

闭包的作用参考：https://www.cnblogs.com/zachary93/p/6056987.html

17、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。

18、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

19、JSON 的了解

JSON(JavaScript Object Notation, JS 对象标记) 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"}

20、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

简单的 ajax 原理分析：

```
//打开浏览器
var xhr = new XMLHttpRequest();
//在地址栏输入地址
xhr.open('get','1.txt',true);
//提交
xhr.send();       //在这一步才会发送网络请求（上面是准备工作）

//等待服务器返回内容
xhr.onreadystatechange = function() {
    if ( xhr.readyState == 4 ) {
        if ( xhr.status == 200 ) {    //200 表示 http 请求正确
```

```
                    alert( xhr.responseText );
                } else {
                    alert('出错了,Err：' + xhr.status);     //返回错误信息
                }
            }

        }
  /**********js 封装好后的 ajax 函数**********/
function ajax(opt) {
    defaultOpt = {   //默认函数参数
    "method":'get', //默认为 get 传输
    "data":'',   //默认没有数据
    "url":'',
    "succeed": function(datas){} //默认获取到数据后什么都不做
    }
   extend(defaultOpt, opt);   //配置参数赋值给默认参数
    var xhr = new XMLHttpRequest();
    if (defaultOpt.method == 'get' && defaultOpt.data) {   //调用默认参数
        defaultOpt.url += '?' + defaultOpt.data;
    }
    xhr.open(defaultOpt.method, defaultOpt.url, true);
    if (defaultOpt.method == 'get') {
        xhr.send();
    } else {
        xhr.setRequestHeader('content-type', 'application/x-www-form-urlencoded');
        xhr.send(defaultOpt.data);
    }

    xhr.onreadystatechange = function () {

        if (xhr.readyState == 4) {
            if (xhr.status == 200) {
                defaultOpt.succeed && defaultOpt.succeed(xhr.responseText);   //可变的地
方用参数传入，代码块就用函数传入

            } else {

                alert('出错了' + xhr.status);
            }

        }

    }
```

```
}
function extend(obj1, obj2) {
    for (var attr in obj2) {
        obj1[attr] = obj2[attr]
    }
}
```

调用形式：

```
ajax({
        "url": 'url 地址',
        "method":'请求方式',
        "data": '传给后台的参数数据',
        "succeed": function (datas) {       //datas 为后台传回来的数据
                //var data = JSON.parse(datas);   //把字符串类数组，转换为对象
            }
    });
```

21、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 后面加上时间戳："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。

22、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

23、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

语法：

```
var e = document.createElement("元素名");          //创建元素节点
var t = document.createTextNode("元素内容");   //创建文本节点
e.appendChild(t);                                  //把元素内容插入元素中去
```

都保存在变量中为了操作方便

（2）添加、移除、替换、插入

appendChild()

语法：

```
obj.appendChild(new)
```

说明：

obj表示当前节点，new表示新节点。

removeChild()

语法：

```
obj.removeChild(oldChild);
```

说明：

参数obj表示当前父节点，而参数oldChild表示需要当前节点内部的某个子节点。

replaceChild()

语法：

```
obj.replaceChild(new,old)
```

说明：

obj，表示被替换节点的父节点；

new，表示替换后的新节点；

old，需要被替换的旧节点。

insertBefore() //在已有的子节点前插入一个新的子节点

语法：

```
obj.insertBefore(new,ref)
```

说明：

obj表示父节点；

new表示新的子节点；

ref指定一个节点，在这个节点前插入新的节点。

（3）查找元素

//通过标签名称　document. getElementsByTagName("标签名")
getElementsByTagName()

//通过 name 属性来获取表单元素(IE 容错能力较强，会得到一个数组，其中包括 id 等于 name 值的)　　//document.getElementsByName("name 属性的属性值")
getElementsByName()

//通过元素 Id，唯一性　　document.getElementById("id 名")
getElementById()

//通过 class 来选中元素
document. getElementsByClassName("类名")

//querySelector()表示选取满足选择条件的第 1 个元素，querySeletorAll()表示选取满足条件的所有元素。　//选择器直接写 css 选择器形式#id
document.querySelector("选择器");
document.querySeletorAll("选择器");

24、js 中某些兼容性的问题的解决方案

1.把 input 元素的 type 属性在发生某事件后变成自己想要的 type 值。

比如在点击单行文本框时，变成复选框。那么在旧版本的 ie 浏览器是不支持的，那么我们

就需要学会一个思想，就是不要直接改 type 值，而是把文本框在点击时隐藏起来，实现隐藏的复选框在点击时出现。
所以说我们为了实现一个效果可以换个思想实现。

2.当我们要改 div 的 float 值时，也存在兼容性问题。

```
oDiv.style.styleFloat = 'left';
oDiv.style.cssFloat = 'left';

IE（styleFloat）、非IE（cssFloat）
```

（出了上面的方法可以解决外，我们还可以用动态添加 class 的方法来达到目的。）

25、[ ]的使用

```
// JS中允许 "." 替换成 "[]"
oBtn.onclick = function (){
    oAttr.value    // 'width'  'background'
    oVal.value     // '200px'  'red'

    // oDiv.style.width = oVal.value;     // . 后面的值无法修改
    oDiv.style[oAttr.value] = oVal.value; // []里面的值可以随便写
};
```

任何地方的.（点）都可以换成[]（中括号）
中括号一般是用在要变的地方，点后面的值是无法改变的
中括号中的值是字符串型则要加引号

26、for 循环的性能问题

```
var str = '';
for( var i=0; i<2000; i++ ){
  // document.body.innerHTML += '<input type="button" value="按钮" />';
  str += '<input type="button" value="按钮" />';
}

document.body.innerHTML = str;
};
```

类似于上面的这种情况，注释掉的那行代码在性能上执行很不好，比较慢，因为每次循环都要到 body 中添加，计算后又循环，于页面交互性太多会导致速度慢。
但是现在上面的方法就换用字符串来存储，存储完 2000 个字符串后再一次性添加到页面中，这样的话与页面的交互性比较少，速度会快很多。

27、js 预解析机制

作用域： 一个 script 标签就是一个作用域，如果一个页面有多个 script 标签则会从上到下执行，先把上面的 script 标签内部代码执行完了，才会执行下面的 script 标签内的代码
（每遇到一个作用域就会进行两步骤）

在浏览器中，对 js 的代码的解析步骤：

1. （找出一些关键字）如：var、函数、参数

（如果找到的是 var 的变量，则统一先把他们存为未定义（undefined），如果找到的是函数，则存为不变的整个函数块 （参数的话其实也是一个 var 变量））

（如果同名：1.一个是变量，一个是函数，那么直接留下函数，而不考虑顺序问题；
2.如果都是函数，那么就会考虑顺序问题，后者留下的原则。）

```
a = 未定义
   所有的变量，在正式运行代码之前，都提前赋了一个值：未定义
fn1 = function fn1(){ alert(2); }
   所有的函数，在正式运行代码之前，都是整个函数块
```

2. 逐行解读代码

1）如果遇到的是表达式， 解读表达式：+ - * / % ++ -- 参数 .....（这些能够改变变量的值的） 把相应的变量值改了
2）如果遇到函数调用，则又要进行预解析的两个步骤！（因为函数又是一个作用域）
（这时是在本函数内部进行预解析）

1.（找出一些关键字）如：var、函数、参数

（如果函数内部没有 var 等关键字，那么内部函数就会到函数外去找（所谓的由内向外））

2.逐行解读代码

例：

```
alert(a);        // function a (){ alert(4); }
var a = 1;
alert(a);        // 1
function a (){ alert(2); }
alert(a);        // 1
var a = 3;
alert(a);        // 3
function a (){ alert(4); }
alert(a);        // 3
```

1.（找出一些关键字）

1) 第一轮 pk　　最后 a = function a() { alert(2);}

a = undefine;

a = function a() { alert(2);}

1) 第二轮 pk　　最后 a = function a() { alert(4);}

a = function a() { alert(2);}

a = function a() { alert(4);}

2.逐行解读代码 （在这步时遇到函数的定义就直接忽略了，因为又不是表达式，又不是函数调用的）

```
alert(a);        // function a (){ alert(4); }
var a = 1;
alert(a);        // 1
function a (){ alert(2); }
alert(a);        // 1
var a = 3;
alert(a);        // 3
function a (){ alert(4); }
alert(a);        // 3
```

当逐行解读代码的时候每一行都会到自己的第一步去找看是否有那个变量，并相应的对变量进行修改（表达式会修改变量）

（最后弹出的值如图的所示）

二、js 没有块作用域，if、for、这些都不是一个作用域

（在写代码的时候最好不要在这些里面声明变量和函数，因为有写浏览器解析不出来，就会直接报错）

```
// alert(a);    // ...
alert( fn1 );  // FF 不能对下面的函数进行预解析

if( true ){
  var a = 1;
  function fn1(){
    alert(123);     // 12345654321345654321
  }
}
```

（最好的做法如下：）

```
// alert( fn1 );   // FF 不能对下面的函数进行预解析

var a = 1;
function fn1(){
  alert(123);
}

if( true ){

}
```

把他们统统放到外边去定义

28、JSONP 解决跨域问题（原理）

原理是：动态插入 script 标签，通过 script 标签引入一个 js 文件，这个 js 文件载入成功后会执行我们在 url 参数中指定的函数，并且会把我们需要的 json 数据作为参数传入。

29、数组去重

1、封装函数法：

2、es6 方式

```
let arr = [1, 2, 2, 3, "a", "a"];
let result = Array.from(new Set(arr));
console.log(result);              //[1, 2, 3, "a"]
```

30、计算一年中有多少周

算法按元旦后第一个星期日才算第一周计算,一年只有 52 或 53 周.

```
function getNumOfWeeks(year){
var d=new Date(year,0,1);
var yt=( ( year%4==0 && year%100!=0) || year%400==0)?366:365;
return Math.ceil((yt-d.getDay())/7.0);
}
var a=[2012,2011,2000,1900];
for(var i in a){
document.write(a[i]+"年有"+getNumOfWeeks(a[i])+"周
");
}
```

31、页面内有一个正方形元素，实现对其拖拽和放下，考虑边界（考虑浏览器兼容性）

```
window.onload = function(){
    var oDiv = document.getElementById('demo');
    //记录鼠标在元素上的位置
    var disX = 0;
    var disY = 0;
    oDiv.onmousedown = function(ev){
        var ev = ev || window.event;
        disX = ev.clientX - oDiv.offsetLeft;
        disY = ev.clientY - oDiv.offsetTop;
        document.onmousemove = function(ev){
            var ev = ev || window.event;
            var L = ev.clientX - disX;
            var T = ev.clientY - disY;
            //可视区的宽度减去自身的宽度(包括 width+padding+border)；可视区的高度减去自身的高度
            var maxL = document.documentElement.clientWidth-oDiv.offsetWidth,
                maxT = document.documentElement.clientHeight-oDiv.offsetHeight;
            if(L < 0){
                L = 0;
            }else if(L >= maxL){
                L = maxL;
            }

            if(T < 0){
                T = 0;
            }else if( T >= maxT){
                T = maxT;
            }
            oDiv.style.left = L + 'px';
            oDiv.style.top = T + 'px';
        };
        document.onmouseup = function(){
            document.onmousemove = null;
            document.onmouseup = null;
        };
        return false;
    };
};
```

四、Web 前端性能优化——如何提高页面加载速度

1、减少 HTTP 请求

用 css sprites 降低图片数量

尽可能使用字体图标，字体图标可以减少很多图片的使用，从而减少 http 请求

将多个样式表或者脚本文件合并到一个文件中，可以减少 HTTP 请求的数量从而缩短效应时间（这个要看情况而定）。

2、将样式表放在头部

将样式表放在头部对于实际页面加载的时间并不能造成太大影响，但是这会减少页面首屏出现的时间，使页面内容逐步呈现，改善用户体验，防止“白屏”。

3、将脚本放在底部

js 的下载和执行会阻塞 Dom 树的构建（严谨地说是中断了 Dom 树的更新），所以 script 标签放在首屏范围内的 HTML 代码段里会截断首屏的内容。

4、使用外部的 JavaScript 和 CSS，缓存

5、css 选择器的使用问题（尽量不用后代选择器，少用子代选择器，最好不要超过三层），dom 操作的优化问题（多次循环添加，不如拿个变量来保存，然后只添加一次）

6、使用构建工具，压缩代码，压缩图片

五、一个页面从输入 URL 到页面加载显示完成的详细过程

1、浏览器查找域名对应的 IP 地址；

2、 浏览器根据 IP 地址与服务器建立 socket 连接（三次握手）；

主机向服务器发送一个**建立连接的请求**（ 您好，我想认识您 ）；
服务器接到请求后发送**同意连接的信号**（ 好的，很高兴认识您 ）；
主机接到同意连接的信号后，再次向服务器发送了**确认信号**（ 我也很高兴认识您），自此，主机与服务器两者建立了连接。

3、浏览器与服务器通信： 浏览器请求，服务器

处理请求；

浏览器根据 URL 内容生成 HTTP 请求，请求中包含请求文件的位置、请求文件的方式等等；

服务器接到请求后，会根据 HTTP 请求中的内容来决定如何获取相应的 HTML 文件；

服务器将得到的 HTML 文件发送给浏览器；

在浏览器还没有完全接收 HTML 文件时便开始渲染、显示网页；

在执行 HTML 中代码时，根据需要，浏览器会继续请求图片、CSS、JavsScript 等文件，过程同请求 HTML ；

4、浏览器与服务器断开连接（四次挥手）。

主机向服务器发送一个断开连接的请求（ 不早了，我该走了 ）；

服务器接到请求后发送确认收到请求的信号（ 知道了 ）；

服务器向主机发送断开通知（ 我也该走了 ）；

主机接到断开通知后断开连接并反馈一个确认信号（ 嗯，好的 ），服务器收到确认信号后断开连接；

六、优雅降级与渐进增强

渐进增强（progressive enhancement）：

针对低版本浏览器进行构建页面，保证最基本的功能，然后再针对高级浏览器进行效果、交互等改进和追加功能达到更好的用户体验。（从被所有浏览器支持的基本功能开始，逐步地添加那些只有新式浏览器才支持的功能，向页面添加无害于基础浏览器的额外样式和功能。当浏览器支持时，它们会自动地呈现出来并发挥作用。）

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。（Web 站点在所有新式浏览器中都能正常工作，如果用户使用的是老式浏览器，则代码会检查以确认它们是否能正常工作。由于 IE 独特的盒模型布局问题，针对不同版本的 IE 的 hack 实践过优雅降级了，为那些无法支持功能的浏览器增加候选方案，使之在旧式浏览器上以某种形式降级体验却不至于完全失效。）

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。

七、http 状态码有那些？分别代表是什么意思？

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

八、sql 注入原理

就是通过把 SQL 命令插入到 Web 表单递交或输入域名或页面请求的查询字符串，最终达到欺骗服务器执行恶意的 SQL 命令。

总的来说有以下几点：

1.永远不要信任用户的输入，要对用户的输入进行校验，可以通过正则表达式，或限制长度，对单引号和双"-"进行转换等。

2.永远不要使用动态拼装 SQL，可以使用参数化的 SQL 或者直接使用存储过程进行数据查询存取。

3.永远不要使用管理员权限的数据库连接，为每个应用使用单独的权限有限的数据库连接。

4.不要把机密信息明文存放，请加密或者 hash 掉密码和敏感的信息。

九、HTTP 和 HTTPS

HTTP 协议（HyperText Transfer Protocol，**超文本传输协议**）是用于从 WWW 服务器传输超文本到本地浏览器的传送协议。它可以使浏览器更加高效，使网络传输减少。它不仅保证计算机正确快速地传输超文本文档，还确定传输文档中的哪一部分，以及哪部分内容首先显示(如文本先于图形)等。

HTTP 是一个应用层协议，由请求和响应构成，是一个标准的客户端服务器模型。HTTP 是一个无状态的协议。

一次 HTTP 操作称为一个事务，其工作过程可分为四步：

1）首先客户机与服务器需要建立连接。只要单击某个超级链接，HTTP 的工作开始。

2）建立连接后，客户机发送一个请求给服务器，请求方式的格式为：统一资源标识符（URL）、协议版本号，后边是 MIME 信息包括请求修饰符、客户机信息和可能的内容。

3）服务器接到请求后，给予相应的响应信息，其格式为一个状态行，包括信息的协议版本号、一个成功或错误的代码，后边是 MIME 信息包括服务器信息、实体信息和可能的内容。

4）客户端接收服务器所返回的信息通过浏览器显示在用户的显示屏上，然后客户机与服务器断开连接。

如果在以上过程中的某一步出现错误，那么产生错误的信息将返回到客户端，有显示屏输出。对于用户来说，这些过程是由 HTTP 自己完成的，用户只要用鼠标点击，等待信息显示就可以了。

超文本传输协议 HTTP 协议被用于在 **Web 浏览器和网站服务器之间传递信息**，HTTP 协议以**明文方式发送内容**，不提供任何方式的数据加密，如果攻击者截取了 Web 浏览器和网站服务器之间的传输报文，就可以直接读懂其中的信息，因此，HTTP 协议不适合传输一些敏感信息，比如：信用卡号、密码等支付信息。

为了解决 HTTP 协议的这一缺陷，需要使用另一种协议：**安全套接字层超文本传输协议 HTTPS**，为了数据传输的安全，HTTPS 在 HTTP 的基础上加入了 SSL 协议，SSL 依靠证书来验证服务器的身份，并为浏览器和服务器之间的通信加密。

一、HTTP 和 HTTPS 的基本概念

HTTP：是互联网上应用最为广泛的一种网络协议，是一个客户端和服务器端请求和应答的标准（TCP），用于从 WWW 服务器传输超文本到本地浏览器的传输协议，它可以使浏览器更加高效，使网络传输减少。

HTTPS：是以安全为目标的 HTTP 通道，简单讲是 HTTP 的安全版，即 HTTP 下加入 SSL 层，HTTPS 的安全基础是 SSL，因此加密的详细内容就需要 SSL。

HTTPS 协议的主要作用可以分为两种：一种是建立一个信息安全通道，来保证数据传输的安全；另一种就是确认网站的真实性。

二、HTTP 与 HTTPS 有什么区别？

HTTP 协议传输的数据都是未加密的，也就是明文的，因此使用 HTTP 协议传输隐私信息非常不安全，为了保证这些隐私数据能加密传输，于是网景公司设计了 **SSL（Secure Sockets Layer）**协议用于对 HTTP 协议传输的数据进行加密，从而就诞生了 HTTPS。简单来说，HTTPS 协议是由 SSL+HTTP 协议构建的可进行加密传输、身份认证的网络协议，要比 http 协议安全。

HTTPS 和 HTTP 的区别主要如下：

1、https 协议需要到 ca 申请证书，一般免费证书较少，因而需要一定费用。

2、http 是超文本传输协议，信息是明文传输，https 则是具有安全性的 ssl 加密传输协议。

3、http 和 https 使用的是完全不同的连接方式，用的端口也不一样，前者是 80，后者是 443。

4、http 的连接很简单，是无状态的；HTTPS 协议是由 SSL+HTTP 协议构建的可进行加密传输、身份认证的网络协议，比 http 协议安全。

十、跨域问题解决办法及原理

同源定义 ：
如果两个页面拥有相同的**协议**（protocol），**端口**（如果指定），和**主机**，那么这两个页面就属于同一个源（origin）。

URL	结果	原因
http://store.company.com/dir2/other.html	Success	
http://store.company.com/dir/inner/another.html	Success	
https://store.company.com/secure.html	Failure	协议不同
http://store.company.com:81/dir/etc.html	Failure	端口不同
http://news.company.com/dir/other.html	Failure	主机不同

方式一：图片 ping 或 script 标签跨域

图片 ping 常用于跟踪用户点击页面或动态广告曝光次数。
script 标签可以得到从其他来源数据，这也是 JSONP 依赖的根据。
缺点：只能发送 Get 请求 ，无法访问服务器的响应文本（单向请求）

方式二：JSONP 跨域

JSONP（JSON with Padding）是数据格式 JSON 的一种“使用模式”，可以让网页从别的网域要数据。根据 XmlHttpRequest 对象受到同源策略的影响，而利用 <script>元素的这个开放策略，网页可以得到从其他来源动态产生的 JSON 数据，而这种使用模式就是所谓的 JSONP。用 JSONP 抓到的数据并不是 JSON，而是任意的 JavaScript，用 JavaScript 解释器运行而不是用 JSON 解析器解析。所有，通过 Chrome 查看所有 JSONP 发送的 Get 请求都是 js 类型，而非 XHR。
缺点：
只能使用 Get 请求
不能注册 success、error 等事件监听函数，不能很容易的确定 JSONP 请求是否失败
JSONP 是从其他域中加载代码执行，容易受到跨站请求伪造的攻击，其安全性无法确保

方式三：CORS(跨域资源共享)

Cross-Origin Resource Sharing（CORS）跨域资源共享是一份浏览器技术的规范，提供了 Web 服务从不同域传来沙盒脚本的方法，以避开浏览器的同源策略，确保安全的跨域数据传输。现代浏览器使用 CORS 在 API 容器如 XMLHttpRequest 来减少 HTTP 请求的风险来源。与 JSONP 不同，CORS 除了 GET 要求方法以外也支持其他的 HTTP 要求。服务器一般需要增加如下响应头的一种或几种：

Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: POST, GET, OPTIONS
Access-Control-Allow-Headers: X-PINGOTHER, Content-Type
Access-Control-Max-Age: 86400
跨域请求默认不会携带 Cookie 信息，如果需要携带，请配置下述参数：

"Access-Control-Allow-Credentials": true
// Ajax 设置
"withCredentials": true

方式四：window.name+iframe

window.name 通过在 iframe（一般动态创建 i）中加载跨域 HTML 文件来起作用。然后，HTML 文件将传递给请求者的字符串内容赋值给 window.name。然后，请求者可以检索 window.name 值作为响应。

iframe 标签的跨域能力；
window.name 属性值在文档刷新后依旧存在的能力（且最大允许 2M 左右）。
每个 iframe 都有包裹它的 window，而这个 window 是 top window 的子窗口。contentWindow 属性返回<iframe>元素的 Window 对象。你可以使用这个 Window 对象来访问 iframe 的文档及其内部 DOM。

```
<!--
 下述用端口
 10000 表示：domainA
 10001 表示：domainB
-->

<!-- localhost:10000 -->
<script>
   var iframe = document.createElement('iframe');
   iframe.style.display = 'none'; //  隐藏

   var state = 0; //  防止页面无限刷新
   iframe.onload = function() {
         if(state === 1) {
               console.log(JSON.parse(iframe.contentWindow.name));
               //  清除创建的 iframe
               iframe.contentWindow.document.write('');
               iframe.contentWindow.close();
               document.body.removeChild(iframe);
         } else if(state === 0) {
               state = 1;
               //  加载完成，指向当前域，防止错误(proxy.html 为空白页面)
```

```
            // Blocked a frame with origin "http://localhost:10000" from accessing a cross-origin frame.
            iframe.contentWindow.location = 'http://localhost:10000/proxy.html';
        }
    };

    iframe.src = 'http://localhost:10001';
    document.body.appendChild(iframe);
</script>

<!-- localhost:10001 -->
<!DOCTYPE html>
...
<script>
    window.name = JSON.stringify({a: 1, b: 2});
</script>
</html>
```

注意：

直接嵌入其他域（localhots:10001）下的 URL 会报错，所以需要加载完成替换为当前域的 URL(localhots:10000)，proxy.html 为空白页面，只为解决该问题；
跨域请求 iframe
重新设置 src（http://localhost:10000/proxy.html）后导致页面不断刷新，所以通过 state 来控制；
全部获取完结果后，清除该 iframe。

方式五：window.postMessage()

HTML5 新特性，可以用来向其他所有的 window 对象发送消息。需要注意的是我们必须要保证所有的脚本执行完才发送 MessageEvent，如果在函数执行的过程中调用了它，就会让后面的函数超时无法执行。

下述代码实现了跨域存储 localStorage

```
<!--
  下述用端口
  10000 表示：domainA
  10001 表示：domainB
-->

<!-- localhost:10000 -->
<iframe src="http://localhost:10001/msg.html" name="myPostMessage" style="display:none;">
</iframe>
```

```
<script>
    function main() {
        LSsetItem('test', 'Test: ' + new Date());
        LSgetItem('test', function(value) {
            console.log('value: ' + value);
        });
        LSremoveItem('test');
    }

    var callbacks = {};
    window.addEventListener('message', function(event) {
        if (event.source === frames['myPostMessage']) {
            console.log(event)
            var data = /^#localStorage#(\d+)(null)?#([\S\s]*)/.exec(event.data);
            if (data) {
                if (callbacks[data[1]]) {
                    callbacks[data[1]](data[2] === 'null' ? null : data[3]);
                }
                delete callbacks[data[1]];
            }
        }
    }, false);

    var domain = '*';
    // 增加
    function LSsetItem(key, value) {
        var obj = {
            setItem: key,
            value: value
        };
        frames['myPostMessage'].postMessage(JSON.stringify(obj), domain);
    }
    // 获取
    function LSgetItem(key, callback) {
        var identifier = new Date().getTime();
        var obj = {
            identifier: identifier,
            getItem: key
        };
        callbacks[identifier] = callback;
        frames['myPostMessage'].postMessage(JSON.stringify(obj), domain);
    }
    // 删除
    function LSremoveItem(key) {
```

```
        var obj = {
            removeItem: key
        };
        frames['myPostMessage'].postMessage(JSON.stringify(obj), domain);
    }
</script>

<!-- localhost:10001 -->
<script>
    window.addEventListener('message', function(event) {
        console.log('Receiver debugging', event);
        if (event.origin == 'http://localhost:10000') {
            var data = JSON.parse(event.data);
            if ('setItem' in data) {
                localStorage.setItem(data.setItem, data.value);
            } else if ('getItem' in data) {
                var gotItem = localStorage.getItem(data.getItem);
                event.source.postMessage(
                    '#localStorage#' + data.identifier +
                    (gotItem === null ? 'null#' : '#' + gotItem),
                    event.origin
                );
            } else if ('removeItem' in data) {
                localStorage.removeItem(data.removeItem);
            }
        }
    }, false);
</script>
```

注意 Safari 下会报错：

Blocked a frame with origin “http://localhost:10001” from accessing a frame with origin “http://localhost:10000“. Protocols, domains, and ports must match.

避免该错误，可以在 Safari 浏览器中勾选开发菜单==>停用跨域限制。或者只能使用服务器端转存的方式实现，因为 Safari 浏览器默认只支持 CORS 跨域请求。

方式六：修改 document.domain 跨子域

前提条件：这两个域名必须属于同一个基础域名!而且所用的协议，端口都要一致，否则无法利用 document.domain 进行跨域，所以只能跨子域

在根域范围内，允许把 domain 属性的值设置为它的上一级域。例如，在”aaa.xxx.com”域内，可以把 domain 设置为 “xxx.com” 但不能设置为 “xxx.org” 或者”com”。

现在存在两个域名 aaa.xxx.com 和 bbb.xxx.com。在 aaa 下嵌入 bbb 的页面，由于其 document.name 不一致，无法在 aaa 下操作 bbb 的 js。可以在 aaa 和 bbb 下通过 js 将 document.name = 'xxx.com';设置一致，来达到互相访问的作用。

方式七：WebSocket

WebSocket protocol 是 HTML5 一种新的协议。它实现了浏览器与服务器全双工通信，同时允许跨域通讯，是 server push 技术的一种很棒的实现。

需要注意：WebSocket 对象不支持 DOM 2 级事件侦听器，必须使用 DOM 0 级语法分别定义各个事件。

方式八：代理

同源策略是针对浏览器端进行的限制，可以通过服务器端来解决该问题

DomainA 客户端（浏览器） ==> DomainA 服务器 ==> DomainB 服务器 ==> DomainA 客户端（浏览器）

十一、对组件化和模块化的理解

模块化中的模块一般指的是 Javascript 模块，比如一个用来格式化时间的模块。
组件则包含了 template、style 和 script，而它的 Script 可以由各种模块组成。比如一个显示时间的组件会调用上面的那个格式化时间的模块。

组件化更多关注的是 UI 部分，你看到的一个管理界面的弹出框，头部，内容区，确认按钮和页脚都可以是个组件，这些组件可以组成一个弹出框组件，跟其他组件组合又是一个新的组件。
模块化侧重于功能或者数据的封装，一组相关的组件可以定义成一个模块，一个暴露了通用验证方法的对象可以定义成一个模块，一个全局的 json 配置文件也可以定义成一个模块。封装隔离来后，更重要的是解决模块间的依赖关系。

从包含关系上讲，组件是模块的子集。

十二、移动端和 PC 端有什么区别

第一： PC 考虑的是**浏览器的兼容性**，而移动端开发考虑的更多的是**手机兼容性**，因为目前不管是 android 手机还是 ios 手机，**一般浏览器使用的都是 webkit 内核**，所以说做移动端开发，更多考虑的应该是**手机分辨率的适配**，和**不同操作系统的略微差异化**。

第二：在**部分事件的处理上，移动端多出来的事件是触屏事件，而缺少的是 hover 事件**。另外包括**移动端弹出的手机键盘的处理**，这样的问题在 PC 端都是遇不到的。

第三： 在布局上，**移动端开发一般是要做到布局自适应**的，我使用的一直是 rem 布局，感觉很好。

第四： 在动画处理上，PC 端由于要考虑 IE 的兼容性，所以通常使用 JS 做动画的通用性会更好一些，但是 CSS3 做了很大的牺牲， 而在**手机端，如果要做一些动画、特效等，第一选择肯定是 CSS3， 既简单、效率又高。**

第五： 微信的一些接口组好能去实现一遍，熟悉一下肯定是有好处的，比如通过微信分享文章，title、description、icon 等图标的配置，这些还是要会的。

第六： 百度地图的一些 API 接口，也得去实现一下，这些对于移动端来说，LBS 是一个非常重要的特性，所以地图这块肯定是要了解的，在加上百度地图这块已经是一个比较成熟的平台了，所以学起来也比较容易。

第七： CSS3 的动画一定要比较熟练，这在移动端用的还是比较多的。

第八：**移动端和 pc 端适用的 js 框架也是不一样**。 一般 pc 端用 jquery，移动端用 zepto，因为移动端的流量还是比较重要的， 所以引入的资源或者插件，能小则小，一个 30k 的资源和一个 80k 的资源，在移动端的差别还是挺大的。
而未压缩的 jquery 是 262kb， 压缩的 jquey 是 83kb，可见两者的差别之大。

第九：最好能掌握一套完整的前端开发架构，比如模块化、打包、压缩、缓存、有条件的还可以做一下自动化测试等等，比较好用的有 fis，另外，想要快速提升自己的前端开发技术，钻研前端架构这块是一个非常好的方向。

第十： **性能优化**，包括**首屏的打开速度、用户响应延迟、渲染性能、动画帧率等在手机上都需要特别注意**。

第十一： 比如在手机上的 300ms 的延迟，这在 PC 端是没有的，如果我们希望做成 webapp，那么自然就不需要这 300ms 的延迟，所以可以使用 hammer-time.js 来移除这 300ms 的延迟。

十三、移动端适配问题

1、先设置 meta，使网页的宽度默认等于屏幕的宽度
<meta name="viewport" content="width=device-width, initial-scale=1,user-scalable=0">
2、使用 rem 做单位来适配，设置 html 根元素的字体大小。

```
var html = document.documentElement;
var hWidth = html.getBoundingClientRect().width;   //获取到 html 的宽度
html.style.fontSize = hWidth/16 + "px";    //设置页面的字体大小，如果手机的分辨率是 320px，那么求得 fontSize 是 20px（可以在页面中看出）；那么 1rem=20px
```

大公司面试题：

1、css 中的包含块

可以决定一个元素的大小和定位的元素，作用是为他内部的后代元素提供一个参考。一个元素的大小和定位往往是由该元素的所在的包含块决定的。可以当包含块的元素一定是 block、inline-block、table-cell 。

2、css 优化，css 放 Body 前和后的区别

推荐放在 head 标签里是因为浏览器代码解析是从上到下的。如果把 css 放在底部，当网速慢时，html 代码加载完成后而 css 没加载完的话，会导致页面没有样式而难以阅读，影响用户体验。所以先加载 css 样式能让页面正常显示。

3、原型、原型链、继承

原型（prototype）：

去改写对象下面公用（不变的）的方法或者属性，让公用的方法或者属性在内存中存在一份，目的是提高性能。

当我们创建一个函数的时候，每个函数会自动生成一个原型（prototype）属性；在函数中就只有这一个原型属性，而这个属性是一个指针，指向一个对象，称为原型对象，原型对象中含有一个 constructor 属性，通过这个属性又可指回函数
当我们向函数中添加属性时，实际上添加到了原型对象之中，当我们用 new 操作符创建新实例时，这个新实例是可以共享原型对象中的属性的
当我们向新实例中添加属性时，属性被保存到了新实例中，当向新实例中添加和原型中一模一样的属性时，这个属性会覆写原型中的属性

原型链：

实例对象与原型之间的连接，用来实现共享属性和继承的
（因为原型链的存在，创建出来的对象能找到原型对象（其实原型也是一个对象）下面的东西（包括属性和方法））

Js 中的面向对象：

其他面向对象语言：面向对象的语言有一个标志，即拥有类(class)的概念，抽象实例对象的公共属性与方法，基于类可以创建任意多个实例对象，一般具有封装、继承、多态的特性！(php/java/c#等)
JS 的对象是一组无序的值，其中的属性或方法都有一个名字，根据这个名字可以访问相映射的值（值可以是基本值/对象/方法）。

1、理解对象（js创建对象一般用两种方法）：

（1）基于Object对象：

```javascript
var stu=new Object();
   stu.name='jack';
   stu.age=20;
  stu.job='worker';
  stu.getjob=function(){return this.job;}
```

（2）基于对象字面量表达式：

```javascript
var stu={
  name:'jack',
  age:20,
  job:'worker',
  getjob:function(){alert(this.job);};
```

对象的继承：（是复用代码的一种形式）

在原有对象的基础上，略加修改，得到一个新的对象，这个新对象继承了原来对象的属性和方法；且不影响原有对象的功能。

拷贝继承：（需要的属性和方法就直接继承过来，不想要继承的功能函数，可以修改子类本身对应的那个函数）

属性的继承：（需要用 call 修改 this 指向）调用父类的构造函数

方法的继承：（解决对象赋值中存在值和引用的赋值） for in 形式

（函数间的赋值如果函数存在同名则是复制的关系，不是直接赋值覆盖）

```
function Child(id) {     //子类
        Father.call(this, id);   //属性的继承
}
extent(Child.prototype, Father.prototype); //方法的继承

function extent(node1, node2) {   //避免对象直接进行赋值，所以拟用对象内的基本数据类型来进行赋值，那么修改子类的函数时，就不会影响到父类（node2 内的方法复制给 node1）
                for (var attr in node2) {
                        node1[attr] = node2[attr];
                }
        }
```

类式继承：利用构造函数（类）继承的方式 (基本原理是通过对象赋值和原型链得到继承关系)

继承的方式：属性和方法要分开继承

属性继承：（需要用 call 修改 this 指向）调用父类的构造函数

方法继承：**子类要继承的构造函数名.prototype = new 父类构造函数名();**

类：js 中是没有类的概念，但是我们可以把构造函数看成是类。

原型继承：借助原型来实现对象的继承

```
var a = {
      name : '小明'
};
var b = cloneObj(a);
b.name = '小强';
//alert( b.name );   //小强
alert( a.name );     //小明
```

```
function cloneObj(obj){
    var F = function(){};
    F.prototype = obj;
    return new F();        //上面这三句话就实现了原型继承
}
```

4、rem 的优缺点

优点：

1、实现强大的屏幕适配布局，rem 是通过根元素进行适配的，网页中的根元素指的是 html 我们通过设置 html 的字体大小就可以控制 rem 的大小。在移动端使用较多。

2、没有屏幕字体缩放问题：

px 像素会在某些浏览器设置页面缩放比的情况下字体不按比例缩放。rem 是相对大小，没有这种问题

3、没有 em 多次使用计算麻烦的问题：

em 是相对于父元素的大小，当层级较多，需要使用时，容易遇到无法预知的风险，也麻烦，而 rem 统一相对根元素，没有这种弊端。

缺点：

1、rem 在多屏幕尺寸适配上与当前两大平台的设计哲学不一致。

iOS 与 Android 平台的适配方式背后隐藏的设计哲学是这样的：阅读文字时，可读性较好的文字字号行距等绝对尺寸数值组合与文字所在媒介的绝对尺寸关系不大。（可以这样简单理解：A4 大小的报纸和 A3 大小甚至更大的报纸，舒适的阅读字号绝对尺寸是一样的，因为他们都需要拿在手里阅读，在手机也是上同理）；在看图片视频时，图片、视频的比例应该是固定的，不应该出现拉伸变形的情况。而 rem 用在字号时，使字号在不同屏幕上的绝对尺寸不一致，违背了设计哲学。

2、图片像素不高的时候，小屏到大屏的放大会让图片变模糊

5、var a = [] 与 var a = new Array()的区别

1、new Array() 会实例化一个对象变量，而 var arr=[],等于是直接声明一个变量。很明显实例一个对象对性能的损耗比直接声明一个对象来的大些，但只有在大批量数据的情况下才会有影响。

2、[ ]和 new Array()在语法上唯一的区别是 new Array()可以直接设置数组的长度

6、["1", "2", "3"].map(parseInt)得到什么？

答案是：[1, NaN, NaN].

原因：parseInt 接收的是两个参数，map 传递的是 3 个参数。

map()方法将调用的数组的每个元素传递给指定的函数，并**返回一个数组**，它包含该函数的返回值。

7、事件代理事件委托

事件冒泡：子级元素的某个事件被触发，它的上级元素的该事件也被递归触发。

事件委托：使用了事件冒泡的原理，从触发某事件的元素开始，递归地向上级元素传播事件。

事件委托的优点：

1) 对于要大量处理的元素，不必为每个元素都绑定事件，只需要在它们的父元素上绑定一次即可，提高性能。

2) 可以处理动态插入 DOM 中的元素，对动态插入 DOM 中的元素进行直接绑定是不行的。

关于事件委托有一个问题：事件委托给父元素后，如何得知事件是哪个子元素触发的？

答：可以通过 event.target 对象来获取。

```
window.onload = function(){
    var oUl = document.getElementById("ul1");
    oUl.onclick = function(ev){
        var ev = ev || window.event;
        //兼容性写法
        var target = ev.target || ev.srcElement;
        if(target.nodeName.toLowerCase() == 'li'){
                alert(123);
                alert(target.innerHTML);
        }
    }
}
```

8、阻止事件冒泡

```
//阻止冒泡事件的兼容性处理
        function stopBubble(e) {
            if(e && e.stopPropagation) { //非 IE
                e.stopPropagation();
            } else { //IE
                window.event.cancelBubble = true;
            }
        }
```

9、http 首部

HTTP 首部字段是构成 HTTP 报文的要素之一。在客户端与服务器之间以 HTTP 协议进行通信的过程中，无论是请求还是响应都会使用首部字段，它能起到传递额外重要信息的作用。使用首部字段是为了给浏览器和服务器提供报文主体大小、所使用的语言、认证信息等内容

可以将首部分为五个主要的类型，包括通用首部、请求首部、响应首部、实体首部和扩展首部

1. 通用首部字段（就是请求报文和响应报文都能用上的字段）

字段名称	说明
Cache-Control	控制缓存的行为
Pragma	http1.0 的旧社会遗留物，值为“no-cache”时禁用缓存

2. 请求首部字段

字段名称	说明
If-Match	比较 ETag 是否一致
If-None-Match	比较 ETag 是否不一致
If-Modified-Since	比较资源最后更新的时间是否一致
If-Unmodified-Since	比较资源最后更新的时间是否不一致

3. 响应首部字段

字段名称	说明
ETag	资源的匹配信息

4. 实体首部字段

字段名称	说明
Expires	http1.0 的遗留物，实体主体过期的时间
Last-Modified	资源的最后一次修改的时间

10、箭头函数和普通函数的区别

箭头函数在定义之后，this 就不会发生改变了，无论用什么样的方式调用它，this 都不会改变；

1、箭头函数的 this 永远指向其上下文的 this，任何方法都改变不了其指向，如 call(), bind(), apply()

普通函数的 this 指向调用它的那个对象

2、箭头函数作为匿名函数,是不能作为构造函数的,不能使用 new

3、箭头函数不绑定 arguments,取而代之用 rest 参数…解决

4、通过 call() 或 apply() 方法调用一个函数时，只是传入了参数而已，对 this 并没有什么影响

5、箭头函数没有原型属性

6、箭头函数不能换行

7、箭头函数不能当做 Generator 函数,不能使用 yield 关键字

11、apply，call，bind 的区别，以及怎么使用，参数问题

call() 、apply()可以看作是某个对象的方法，通过调用方法的形式来间接调用函数（改变函数内部 this 的指向）。bind() 就是将某个函数绑定到某个对象上。

call()在第一个参数之后的 后续所有参数就是传入该函数的值。就是说要全部列举出来。

apply() 只有两个参数，第一个是对象，第二个是数组，这个数组就是该函数的参数。

this 指向他们的第一个参数

bind（）--也是改变函数体内 this 的指向;

bind 会创建一个新函数，称为绑定函数，当调用这个函数的时候，绑定函数会以创建它时传入 bind（）方法的第一个参数作为 this，传入 bind（）方法的第二个及以后的参数加上绑定函数运行时本身的参数按照顺序作为原函数的参数来调用原函数；

bind 与 apply、call 最大的区别就是：bind 不会立即调用，其他两个会立即调用

12、页面重绘（repaint）和页面回流（reflow）

css 与 dom 解析完毕后，合并为渲染树（render tree）。

重绘(repaint)：当 render tree 中的一些元素需要更新属性，单这些属性只会影响元素的

外观，风格，而不会影响到元素的布局，此类的页面渲染叫作页面重绘。

回流(reflow)：当 render tree 中的一部分（或全部）因为元素的规模尺寸，布局，隐藏等改变而引起的页面重新渲染。

可以看出，回流必将引起重绘，而重绘不一定会引起回流。

当页面布局和几何属性改变时就需要回流。下述情况会发生浏览器回流：

DOM 树发生变化——如：增加一个元素或删除一个元素（元素为可见元素）；
通过 style 控制元素的位置变化——典型的碰壁反弹；
元素尺寸的改变——盒模型的每种尺寸均算在其内；
内容改变引发的尺寸改变——如：文本文本改变或者图片大小改变而引起的计算值宽度和高度改变；
浏览器窗口尺寸改变——resize 事件发生时。

改善由于 dom 操作产生的回流：

1、不要一个一个改变元素的样式属性，直接改变 className，如果动态改变样式，则使用 cssText。

2、使用 DocumentFragment 进行缓存操作,引发一次回流和重绘；

这个主要用于添加元素的时候，就是先把所有要添加到元素添加到 1 个 div 中(这个 div 也是新加的)，最后才把这个 div append 到 body 中。

3、使用 display:none 技术，只引发两次回流和重绘；

先 display:none 隐藏元素，然后对该元素进行所有的操作，最后再显示该元素。因对 display:none 的元素进行操作不会引起回流、重绘。所以只要操作只会有 2 次回流。

4、使用 cloneNode(true or false) 和 replaceChild 技术，引发一次回流和重绘。

13、字符串反转

```
var str = "a,b,c,d,e,s";
function reverseString(str) {
    return str.split('').reverse().join('');
}
console.log(reverseString(str));    //s,e,d,c,b,a
```

相关知识点：

split() 方法用于把一个**字符串**分割成**字符串数组**。

reverse() 方法用于颠倒**数组**中元素的顺序,并且会**直接修改原数组**。

方法用于把数组中的**所有元素**放入一个字符串,并按传入的分隔符进行分割。

14、ES6 十大特性

1.Default Parameters（默认参数） in ES6

还记得我们以前(ES5)不得不通过下面方式来定义默认参数：

```javascript
var link = function (height, color, url) {
  var height = height || 50;
  var color = color || 'red';
  var url = url || 'http://azat.co';
    ...
}
```

一切工作都是正常的，直到参数值是0后，就有问题了，因为在JavaScript中，0表示fasly，它是默认被hard-coded的值，而不能变成参数本身的值。

当然，如果你非要用0作为值，我们可以忽略这一缺陷并且使用逻辑OR就行了！

但在ES6，我们可以直接把默认值放在函数申明里：

```javascript
var link = function(height = 50, color = 'red', url = 'http://azat.co') {
  ...
}
```


2.Template Literals（模板对象） in ES6

在其它语言中，使用模板和插入值是在字符串里面输出变量的一种方式。

因此，在ES5，我们可以这样组合一个字符串：

```javascript
var name = 'Your name is ' + first + ' ' + last + '.';
var url = 'http://localhost:3000/api/messages/' + id;
```

幸运的是，在ES6中，我们可以使用新的语法$ {NAME}，并把它放在反引号里：

```javascript
var name = `Your name is ${first} ${last}. `;
var url = `http://localhost:3000/api/messages/${id}`;
```

3.Multi-line Strings （多行字符串）in ES6

在ES5中，我们不得不使用以下方法来表示多行字符串：

```javascript
var roadPoem = 'Then took the other, as just as fair,nt'
  + 'And having perhaps the better claimnt'
  + 'Because it was grassy and wanted wear,nt'
  + 'Though as for that the passing therent'
  + 'Had worn them really about the same,nt';

var fourAgreements = 'You have the right to be you.n
  You can only be you when you do your best.';
```

然而在ES6中，仅仅用反引号就可以解决了：

```javascript
var roadPoem = `Then took the other, as just as fair,
  And having perhaps the better claim
  Because it was grassy and wanted wear,
  Though as for that the passing there
  Had worn them really about the same,`;

var fourAgreements = `You have the right to be you.
  You can only be you when you do your best.`;
```

15、懒加载的原理及实现

懒加载的原理：

原理：先将 img 标签中的 src 链接设为同一张图片（空白图片），将其真正的图片地址存储再 img 标签的自定义属性中（比如 data-src）。当 js 监听到该图片元素进入可视窗口时，即将自定义属性中的地址存储到 src 属性中，达到懒加载的效果。

这样做能防止页面一次性向服务器响应大量请求导致服务器响应慢，页面卡顿或崩溃等问题。

1、既然懒加载的原理是基于判断元素是否出现在窗口可视范围内，首先我们写一个函数判断元素是否出现在可视范围内：

```
function isVisible($node){
    var winH = $(window).height(),
        scrollTop = $(window).scrollTop(),
        offSetTop = $($node).offSet().top;
    if (offSetTop < winH + scrollTop) {      //元素到可视窗口顶部的距离 < window 高度+滚动条高度的距离时，说明元素已经进入了可视区范围。
        return true;
    } else {
        return false;
    }
}
```

2、 再添加上浏览器的事件监听函数，让浏览器每次滚动就检查元素是否出现在窗口可视范围内：

```
$(window).on("scroll", function{
    if (isVisible($node)){
        console.log(true);
    }
})
```

3、我们已经很接近了，现在我们要做的是，让元素只在第一次被检查到时打印 true，之后就不再打印了

```
var hasShowed = false;
$(window).on("sroll",function{
    if (hasShowed) {
```

```
            return;
        } else {
            if (isVisible($node)) {
                hasShowed = !hasShowed;
                console.log(true);
            }
        }
    })
```

16、CSS 预处理器的优缺点有哪些

缺点：简单来说 CSS 预处理器语言较 CSS 玩法变得更高级了，但同时降低了自己对最终代码的控制力。更致命的是提高了门槛，首先是上手门槛，其次是维护门槛，再来是团队整体水平和规范的门槛。这也造成了初学学习成本的昂贵。

优点：用一种专门的编程语言，为 CSS 增加了一些编程的特性，减少冗余代码，提高样式代码的可维护性。

17、Bootstrap 响应式栅格系统的设计原理

Bootstrap 采取 12 列的栅格体系，根据主流设备的尺寸进行分段，每段宽度固定，通过百分比和媒体查询实现响应式布局。

首先通过媒体查询确认 container 的宽度，每个 col-xx-xx 都是通过百分比定义的，屏幕尺寸变化了，container 就变化了，col 自然就变了啊

```css
.container {
  padding-right: 15px;
  padding-left: 15px;
  margin-right: auto;
  margin-left: auto;
}
@media (min-width: 768px) {
  .container {
    width: 750px;
  }
}
@media (min-width: 992px) {
  .container {
    width: 970px;
  }
}
@media (min-width: 1200px) {
  .container {
    width: 1170px;
  }
}
```

```
.col-xs-1, .col-xs-2, .col-xs-3, .col-xs-4, .col-xs-5, .col-xs-6,
.col-xs-7, .col-xs-8, .col-xs-9, .col-xs-10, .col-xs-11, .col-xs-12 {
  float: left;
}
.col-xs-12 {
  width: 100%;
}
.col-xs-11 {
  width: 91.66666667%;
}
.col-xs-10 {
  width: 83.33333333%;
}
.col-xs-9 {
  width: 75%;
}
.col-xs-8 {
  width: 66.66666667%;
}
.col-xs-7 {
  width: 58.33333333%;
}
```

Es6：一些箭头函数和 promise 很重要！！！

为什么在 ES6 中引入了箭头函数呢？ 最主要的目的就是解决 this 指针的问题。

怎么学前端的？看过哪些书？逛了哪些社区，或者去哪些学习平台？

前端的学习途径一定要回答具体，比如网站类有妙味，慕课网，绿叶，博客有博客园，张鑫旭，廖雪峰，伯乐在线，思否（SegmentFault）、CSDN 博客

公众号有前端大全，css 精品，前端早读课，伯乐在线，书也看了很多，参考：https://www.zhihu.com/question/22591993

《css 权威指南》、《JavaScript 权威指南》、《JavaScript 高级程序设计》、《JavaScript DOM 编程艺术》、《JavaScript 编程精解》、《JQuery 权威指南》、《ECMAScript6 入门》

http://es6.ruanyifeng.com/#docs/intro =》 《ECMAScript6 入门》

20、vue 双向绑定的实现原理

21、实现三栏布局，中间 200px，两边自适应

22、让一个元素不可见的方法有哪些

```
{ display: none; /* 不占据空间，无法点击 */ }
/**************************************************************************
*****/
{ visibility: hidden; /* 占据空间，无法点击 */ }
/**************************************************************************
*****/
{ position: absolute; top: -9999px; /* 不占据空间，无法点击 */ }
/**************************************************************************
*****/
{ position: relative; top: -9999px; /* 占据空间，无法点击 */ }
/**************************************************************************
*****/
{ position: absolute; visibility: hidden; /* 不占据空间，无法点击 */ }
/**************************************************************************
*****/
{ height: 0; overflow: hidden; /* 不占据空间，无法点击 */ }
/**************************************************************************
*****/
{ opacity: 0; filter:Alpha(opacity=0); /* 占据空间，可以点击 */ }
/**************************************************************************
*****/
```

```
{ position: absolute; opacity: 0; filter:Alpha(opacity=0); /* 不占据空间，可以点击 */ }
```

23、MVC 与 MVVM 区别？

在 MVC 里，View 是可以直接访问 Model 的！从而，View 里会包含 Model 信息，不可避免的还要包括一些业务逻辑。 MVC 模型关注的是 Model 的不变，所以，在 MVC 模型里，Model 不依赖于 View，但是 View 是依赖于 Model 的。不仅如此，因为有一些业务逻辑在 View 里实现了，导致要更改 View 也是比较困难的，至少那些业务逻辑是无法重用的。

MVVM 在概念上是真正将页面与数据逻辑分离的模式，它把数据绑定工作放到一个 JS 里去实现，而这个 JS 文件的主要功能是完成数据的绑定，即把 model 绑定到 UI 的元素上。

有人做过测试：使用 Angular（MVVM）代替 Backbone（MVC）来开发，代码可以减少一半。

此外，MVVM 另一个重要特性，双向绑定。它更方便你同时维护页面上都依赖于某个字段的 N 个区域，而不用手动更新它们。

24、jQuery 和 vue 有哪些区别，分别使用场景

jquery：使用选择器（$）选取 DOM 对象，对其进行赋值、取值、事件绑定等操作，其实和原生的 HTML 的区别只在于可以更方便的选取和操作 DOM 对象，而数据和界面是在一起的。

vue：通过 Vue 对象将数据和 View 完全分离开来了。对数据进行操作不再需要引用相应的 DOM 对象，可以说数据和 View 是分离的，他们通过 Vue 对象这个 vm 实现相互的绑定。这就是传说中的 MVVM。

vue 适用的场景：复杂的单应用页面；复杂数据操作的后台页面，表单填写页面，侧重数据绑定。

jquery 适用的场景：比如说一些 html5 的动画页面，一些需要 js 来操作页面样式的页面，侧重样式操作，动画效果等。

25、window.onload 何时执行，浏览器如何渲染

window.onload 顾名思义 在页面加载的后执行包括文档，css，script，img，以及 css 里的 background，这个函数是确保页面所有元素加载完成时执行

26、对 seo 的理解和使用

1、title：只强调重点即可，重要关键词出现不要超过 2 次，而且要靠前，每个页面 title 要有所不同

2、description：把网页内容高度概括到这里，长度要合理，不可过分堆砌关键词，每个页面 description 要有所不同
3、keywords：列举出几个重要关键词即可，也不可过分堆砌。
4、语义化书写 HTML 代码，符合 W3C 标准。
对于搜索引擎来说，最直接面对的就是网页 HTML 代码，如果代码写的语义化，搜索引擎就会很容易的读懂该网页要表达的意思。例如文本模块要有大标题，合理利用 h1-h6，列表形式的代码使用 ul 或 ol，重要的文字使用 strong 等等。
5、利用布局，把重要内容 HTML 代码放在最前。
搜索引擎抓取 HTML 内容是从上到下，利用这一特点，可以让主要代码优先读取，广告等不重要代码放在下边。
6、布局层次不要太深，一般重要内容控制在三层之内。
7、重要内容不要用 JS 输出。
蜘蛛不会读取 JS 里的内容，所以重要内容必须放在 HTML 里。
8、尽少使用 iframe 框架。
搜索引擎不会抓取到 iframe 里的内容，重要内容不要放在框架中。
9、为图片加上 alt 属性。
当网络速度很慢，或者图片地址失效的时候，就可以体现出 alt 属性的作用，他可以让用户在图片没有显示的时候知道这个图片的作用。
10、保留文字效果。
如果需要兼顾用户体验和 SEO 效果，在必须用图片的地方，例如个性字体的标题，我们可

以利用样式控制，让文本文字不会出现在浏览器界面上，但在网页代码中是有该标题的文字。
样式控制： overflow:hidden; text-indent:-9999px;

27、JS 中的 Navigator 对象

Navigator 对象包含有关浏览器的信息。

很多时候我们需要在判断网页所处的浏览器和平台，Navigator 为我们提供了便利

Navigator常见的对象属性如下：

属性	描述
appCodeName	返回浏览器的代码名。
appMinorVersion	返回浏览器的次级版本。
appName	返回浏览器的名称。
appVersion	返回浏览器的平台和版本信息。
browserLanguage	返回当前浏览器的语言。
cookieEnabled	返回指明浏览器中是否启用 cookie 的布尔值。
cpuClass	返回浏览器系统的 CPU 等级。
onLine	返回指明系统是否处于脱机模式的布尔值。
platform	返回运行浏览器的操作系统平台。
systemLanguage	返回 OS 使用的默认语言。
userAgent	返回由客户机发送服务器的 user-agent 头部的值。
userLanguage	返回 OS 的自然语言设置。

我们使用的比较多的是他的 userAgent，经常需要判断的情况有：

（1）PC 还是移动端

（2）安卓还是 IOS

（3）浏览器的类型

28、弹性盒子模型

弹性盒模型可以用简单的方式满足很多常见的复杂的布局需求。它的优势在于开发人员只是声明布局应该具有的行为，而不需要给出具体的实现方式。浏览器会负责完成实际的布局。该布局模型在主流浏览器中都得到了支持。
引入弹性盒布局模型的目的是提供一种更加有效的方式来对一个容器中的条目进行排列、对齐和分配空白空间。即便容器中条目的尺寸未知或是动态变化的，弹性盒布局模型也能正常的工作。在该布局模型中，容器会根据布局的需要，调整其中包含的条目的尺寸和顺序来最好地填充所有可用的空间。当容器的尺寸由于屏幕大小或窗口尺寸发生变化时，其中包含的条目也会被动态地调整。比如当容器尺寸变大时，其中包含的条目会被拉伸以占满多余的空白空间；当容器尺寸变小时，条目会被缩小以防止超出容器的范围。弹性盒布局是与方向无关的，可由开发人员操作。

29、git 和 svn 的区别

1.git 是分布式的 scm,svn 是集中式的。(最核心)

分布式：因为每一个开发人员的电脑上都有一个本地仓库,所以即使没有网络也一样可以 Commit，查看历史版本记录，创建项 目分支等操作，等网络再次连接上 Push 到 Server 端。

集中式：SVN 只能有一个指定中央版本库。当这个中央版本库有问题时，所有工作成员都一起瘫痪直到版本库维修完毕或者新的版本库设立完成。

2.git 是每个历史版本都存储完整的文件,便于恢复,svn 是存储差异文件,历史版本不可恢复。(核心)

3.git 可离线完成大部分操作,svn 则不能。

4.git 有着更优雅的分支和合并实现。

5.git 有着更强的撤销修改和修改历史版本的能力

6.git 速度更快,效率更高。

基于以上区别,git 有了很明显的优势,特别在于它具有的本地仓库。

30、浏览器的 http 缓存机制

浏览器为什么要缓存？什么会缓存下来？

首先当我们访问网页的时候，很多大的图片从服务器上传输过来的时候，试想一下，如果浏览器不把图片缓存下来而是每次都要到服务器去取，那么每次都给服务器和网络造成了巨大的负担。

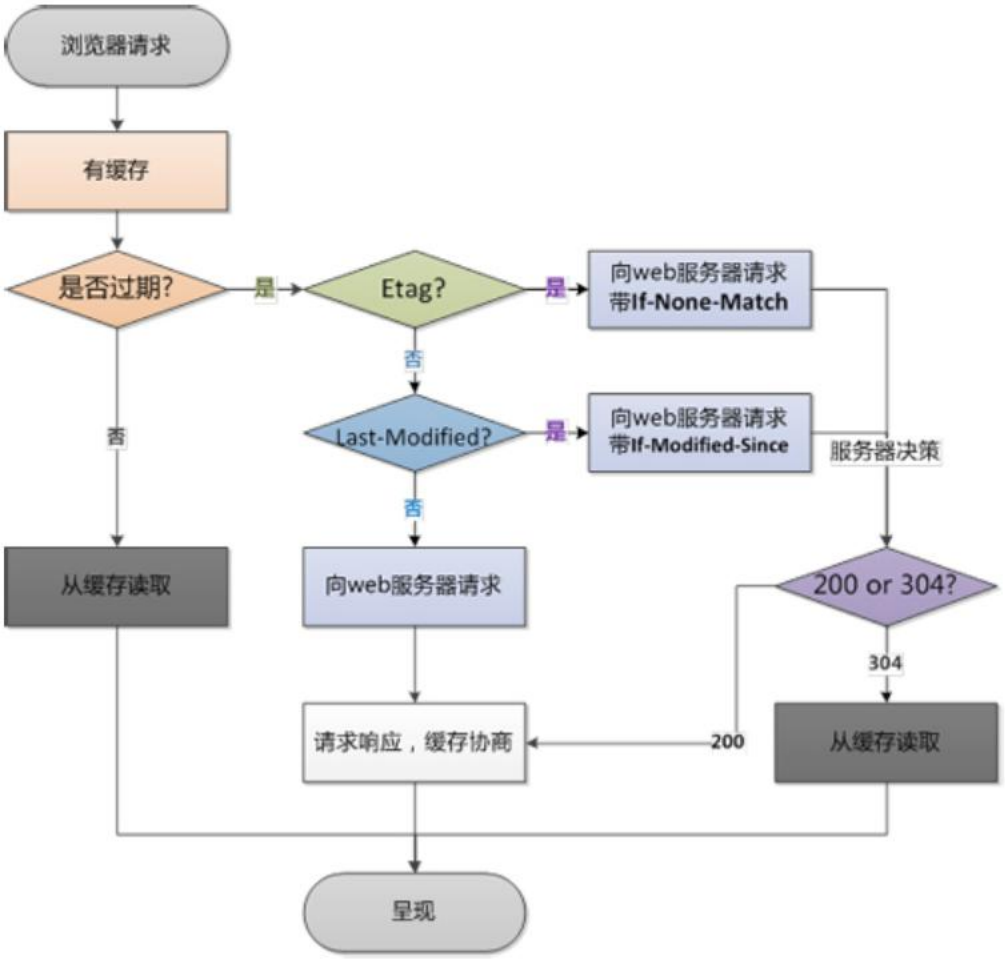
对于静态资源来说，浏览器不会缓存 html 页面的，所以你每次改完 html 的页面的时候，html 都是改完立即生效的，不存在什么有缓存导致页面不对的问题。浏览器缓存的东西有图片，css 和 js。这些资源将在缓存失效前调用的时候调用浏览器的缓存内容。

http 缓存是基于 HTTP 协议的浏览器文件级缓存机制。即针对文件的重复请求情况下，浏览器可以根据协议头判断从服务器端请求文件还是从本地读取文件，以下是浏览器缓存的整个机制流程。主要是针对重复的 http 请求，在有缓存的情况下判断过程主要分 3 步：

◆判断 expires，如果未过期，直接读取 http 缓存文件，不发 http 请求，否则进入下一步。

◆判断是否含有 etag（Etag 由服务器端生成），有则带上 if-none-match 发送请求，未修改返回 304，修改返回 200，否则进入下一步。

◆判断是否含有 last-modified，有则带上 if-modified-since 发送请求，无效返回 200，有效返回 304，否则直接向服务器请求。



如果通过 etag 和 last-modified 判断，即使返回 304 有至少有一次 http 请求，只不过返回的是 304 的返回内容，而不是文件内容。所以合理设计实现 expires 参数可以减少较多的浏览器请求。

用户行为

最后附上一张，用户行为影响浏览器的缓存行为。

用户操作	Expires/Cache-Control	Last-Modified/Etag
地址栏回车	有效	有效
页面链接跳转	有效	有效
新开窗口	有效	有效
前进、后退	有效	有效
F5刷新	无效	有效
Ctrl+F5刷新	无效	无效

31、禁止 http 缓存

Cache-Control

Http1.1 中的标准，可以看成是 expires 的补充。使用的是相对时间的概念。

简单介绍下 Cache-Control 的属性设置：
1）**max-age**: 设置缓存的最大的有效时间，单位为秒（s）。max-age 会覆盖掉 Expires
2) s-maxage: 只用于共享缓存，比如 CDN 缓存（s -> share）。与 max-age 的区别是：max-age 用于普通缓存，
而 s-maxage 用于代理缓存。如果存在 s-maxage,则会覆盖 max-age 和 Expires.
3) **public**：响应会被缓存，并且在多用户间共享。默认是 public。
4) **private**: 响应只作为私有的缓存，不能在用户间共享。如果要求 HTTP 认证，响应会自动设置为 private。
5）**no-cache**: 指定不缓存响应，表明资源不进行缓存。但是设置了 no-cache 之后并不代表浏览器不缓存，而是在缓存前要向服务器确认资源是否被更改。因此有的时候只设置 no-cache 防止缓存还是不够保险，还可以加上 private 指令，将过期时间设为过去的时间。
6）**no-store**: 绝对禁止缓存。
7）must-revalidate: 如果页面过期，则去服务器进行获取。

1、如果需要在 html 页面上设置不缓存，这在<head>标签中加入如下语句：
//是用于设定禁止浏览器从本地机的缓存中调阅页面内容，设定后一旦离开网页就无法从 Cache 中再调出；
<meta http-equiv="pragma" content="no-cache">

//指定不缓存响应，表明资源不进行缓存。但是设置了 no-cache 之后并不代表浏览器不缓存，而是在缓存前要向服务器确认资源是否被更改。
<meta http-equiv="cache-control" content="no-cache, no-store">

//可以用于设定网页的到期时间，一旦过期则必须到服务器上重新调用。需要注意的是必须使用 GMT 时间格式；
<meta http-equiv="expires" content="0">

2、url 后面添加参数，给文件后面设置时间戳，这样每次访问到的就不是同一文件，就没有缓存了。

32、对前端开发的理解

Web 前端开发简单来说就是网站的开发。最基础的工作就是实现效果图，把视觉稿通过页面代码的方式表现出来能够真实反映视觉稿且能够通 过浏览器的兼容。与设计师和后台人员进行沟通，优化和完善前端项目。
作为一名前端开发人员，必须掌握基本的 Web 前端开发技术，其中包括：CSS(3)、HTML(5)、DOM、javascript、Ajax，jquery 等，了解其他框架如 angular、vue、react 等，适应前端技术发展，在掌握这些技术的同时，还要清楚地了解它们在不同浏览器上的兼容情况、渲染原理和存在的 Bug。需要了解网站性能优化和一些打包工具如 webpack、gulp，了解 SEO 等等一

下辅助开发的知识。

33、webpack 和 gulp 区别（模块化与流的区别）

gulp 类似于管道，文件读进来，执行相应的任务然后生成最终文件即可。
gulp 利用流的方式进行文件的处理，通过管道将多个任务和操作连接起来，因此只有一次 I/O 的过程，流程更清晰，更纯粹。

gulp 强调的是前端开发的工作流程，我们可以通过配置一系列的 task，定义 task 处理的事务（例如文件压缩合并、雪碧图、启动 server、版本控制等），然后定义执行顺序，来让 gulp 执行这些 task，从而构建项目的整个前端开发流程。

webpack 是一个前端模块化方案，更侧重模块打包，我们可以把开发中的所有资源（图片、js 文件、css 文件等）都看成模块，通过 loader（加载器）和 plugins（插件）对资源进行处理，打包成符合生产环境部署的前端资源。

34、dom 是什么

DOM （文档对象模型）就是针对 HTML 和 XML 提供的一个 API（接口）。我们可以通过 DOM 来操作页面中各种元素，例如添加元素、删除元素、替换元素等。

35、dom 的 api 有什么

节点查找API

`document.getElementById` ：根据ID查找元素，大小写敏感，如果有多个结果，只返回第一个；

`document.getElementsByClassName` ：根据类名查找元素，多个类名用空格分隔，返回一个 `HTMLCollection` 。注意兼容性为IE9+（含）。另外，不仅仅是document，其它元素也支持 `getElementsByClassName` 方法；

`document.getElementsByTagName` ：根据标签查找元素， `*` 表示查询所有标签，返回一个 `HTMLCollection` 。

`document.getElementsByName` ：根据元素的name属性查找，返回一个 `NodeList` 。

`document.querySelector` ：返回单个Node，IE8+(含），如果匹配到多个结果，只返回第一个。

`document.querySelectorAll` ：返回一个 `NodeList` ，IE8+(含）。

`document.forms` ：获取当前页面所有form，返回一个 `HTMLCollection` ；

3. 节点创建API

节点创建API主要包括 `createElement` 、 `createTextNode` 、 `cloneNode` 和 `createDocumentFragment` 四个方法。

4. 节点修改API

节点修改API都具有下面这几个特点：

1. 不管是新增还是替换节点，如果其原本就
2. 修改之后节点本身绑定的事件不会消失；

4.1. appendChild

这个其实前面已经多次用到了，语法就是：

```
parent.appendChild(child);
```

它会将child追加到parent的子节点的最后
点将会添加到新的位置，其原本所在的位置将
留。

4.2. insertBefore

5. 节点关系API

DOM中的节点相互之间存在着各种各样的关系，如父子关系，兄弟关系等。

5.1. 父关系API

- `parentNode` ：每个节点都有一个parentNode属性，它表示元素的父节点。Element的父节点可能是Element，Document或DocumentFragment；
- `parentElement` ：返回元素的父元素节点，与parentNode的区别在于，其父节点必须是一个Element元素，如果不是，则返回null；

5.2. 子关系API

- `children` ：返回一个实时的 `HTMLCollection` ，子节点都是Element，IE9以下浏览器不支持；
- `childNodes` ：返回一个实时的 `NodeList` ，表示元素的子节点列表，注意子节点可能包含文本节点、注释节点等；
- `firstChild` ：返回第一个子节点，不存在返回null，与之相对应的还有一个 `firstElementChild` ；
- `lastChild` ：返回最后一个子节点，不存在返回null，与之相对应的还有一个 `lastElementChild` ；

5.3. 兄弟关系型API

- `previousSibling` ：节点的前一个节点，如果不存在则返回null。注意有可能拿到的节点是文本节点或注释节点，与预期的不符，要进行处理一下。

36、post 和 get 的区别

GET后退按钮/刷新无害，POST数据会被重新提交（浏览器应该告知用户数据会被重新提交）。
GET书签可收藏，POST为书签不可收藏。
GET能被缓存，POST不能缓存 。
GET编码类型application/x-www-form-url，POST编码类型encodedapplication/x-www-form-urlencoded 或 multipart/form-data。为二进制数据使用多重编码。
GET历史参数保留在浏览器历史中。POST参数不会保存在浏览器历史中。
GET对数据长度有限制，当发送数据时，GET 方法向 URL 添加数据；URL 的长度是受限制的（URL 的最大长度是 2048 个字符）。POST无限制。
GET只允许 ASCII 字符。POST没有限制。也允许二进制数据。
与 POST 相比，GET 的安全性较差，因为所发送的数据是 URL 的一部分。在发送密码或其他敏感信息时绝不要使用 GET ！ POST 比 GET 更安全，因为参数不会被保存在浏览器历史或 web 服务器日志中。
GET的数据在 URL 中对所有人都是可见的。POST的数据不会显示在 URL 中。

37、Webpack 打包（编译加速）

38、移动端的性能优化

移动H5前端性能优化指南 v1.0

PC优化手段在Mobile侧同样适用

在Mobile侧我们提出三秒种渲染完成首屏指标

首屏加载3秒完成或使用Loading

基于联通3G网络平均338KB/s(2.71Mb/s)，所以首屏资源不应超过1014KB

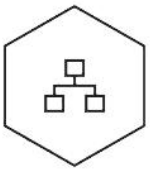
Mobile侧因手机配置原因，除加载外渲染速度也是优化重点

加载优化

1. 合并CSS、JavaScript
2. 合并小图片，使用雪碧图
3. 缓存一切可缓存的资源
4. 使用长Cache
5. 使用外联式引用CSS、JavaScript
6. 压缩HTML、CSS、JavaScript
7. 启用GZip
8. 使用首屏加载
9. 使用按需加载
10. 使用滚屏加载
11. 通过Media Query加载
12. 增加Loading进度条
13. 减少Cookie
14. 避免重定向
15. 异步加载第三方资源

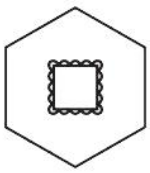
图片优化

1. 使用智图 http://zhitu.tencent.com/
2. 使用（CSS3、SVG、IconFont）代替图片
3. 使用Srcset
4. webP优于JPG
5. PNG8优于GIF
6. 首次加载不大于1014KB（基于3秒联通平均网速所能达到值）
7. 图片不宽于640



脚本优化

1. 减少重绘和回流
2. 缓存Dom选择与计算
3. 缓存列表.length
4. 尽量使用事件代理，避免批量绑定事件
5. 尽量使用ID选择器
6. 使用touchstart、touchend代替click



CSS优化

1. CSS写在头部，JavaScript写在尾部或异步
2. 避免图片和iFrame等的空Src
3. 尽量避免重设图片大小
4. 图片尽量避免使用DataURL
5. 尽量避免写在HTML标签中写Style属性
6. 避免CSS表达式
7. 移除空的CSS规则
8. 正确使用Display的属性
9. 不滥用Float
10. 不滥用Web字体
11. 不声明过多的Font-size
12. 值为0时不需要任何单位
13. 标准化各种浏览器前缀
14. 避免让选择符看起来像正则表达式

渲染优化

1. HTML使用Viewport
2. 减少Dom节点
3. 尽量使用CSS3动画
4. 合理使requestAnimationFrame动画代替setTimeout
5. 适当使用Canvas动画
6. Touchmove、Scroll 事件会导致多次渲染
7. 使用（CSS3 transitions、CSS3 3D transforms、Opacity、Canvas、WebGL、Video）来触发GPU渲染

39、web 移动前端的 click 点透问题

一、点透现象

遮罩层 A 下面有 B 元素，B 元素内有 a 链接，当点击 A 遮罩元素时 A 消失。
当点击 A 遮罩层就在链接 a 的上面时，A 消失的同时链接也跳转了。那么就是 click 点透了。

二、点透出现的场景

刚才举例说明了什么是点透，其实点透的出现场景可以总结如下：

1.A/B 两个层上下 z 轴重叠。

2.上层的 A 点击后消失或移开。（这一点很重要）

3.B 元素本身有默认 click 事件（如 a 标签） 或 B 绑定了 click 事件。

在以上情况下，点击 A/B 重叠的部分，就会出现点透的现象。
欢迎加入全栈开发交流群一起学习交流：864305860

二、为什么会出现点透

click 延迟，延迟，还是延迟。

在移动端不使用 click 而用 touch 事件代替触摸是因为 click 事件有着明显的延迟，具体 touchstart 与 click 的区别如下：

1.touchstart：在这个 DOM（或冒泡到这个 DOM）上手指触摸开始即能立即触发
2.click：在这个 DOM（或冒泡到这个 DOM）上手指触摸开始，且手指未曾在屏幕上移动（某些浏览器允许移动一个非常小的位移值），且在这个在这个 dom 上手指离开屏幕，且触摸和离开屏幕之间的间隔时间较短（某些浏览器不检测间隔时间，也会触发 click）才能触发

也就是说，事件的触发时间按由早到晚排列为：touchstart 早于 touchend 早于 click。亦即 click 的触发是有延迟的，这个时间大概在 300ms 左右。

由于我们在 touchstart 阶段就已经隐藏了罩层 A，当 click 被触发时候，能够被点击的元素则是其下的 B 元素，根据 click 事件的触发规则：

只有在被触发时，当前有 click 事件的元素显示，且在面朝用户的最前端时，才触发 click 事件。

由于 B 绑定了 click 事件（或者 B 本身默认存在 click 事件），所以 B 的 click 事件被触发，产生了点透的情况。

欢迎加入全栈开发交流群一起学习交流：864305860

三、解决方案

1.对于 B 元素本身没有默认 click 事件的情况（无 a 标签等），应统一使用 touch 事件，统一代码风格，并且由于 click 事件在移动端的延迟要大很多，不利于用户体验，所以关于触摸事件应尽量使用 touch 相关事件。

2.对于 B 元素本身存在默认 click 事件的情况,应及时取消 A 元素的默认点击事件，从而阻止 click 事件的产生。即应在上例的 handle 函数中添加代码如下：

```
if(eve == "touchend") e.preventDefault();
```

3.对于遮盖浮层，由于遮盖浮层的点击即使有小延迟也是没有关系的，反而会有疑似更好的用户体验，所以这种情况，可以针对遮盖浮层自己采用 click 事件，这样就不会出现点透问题。

40、伪类和伪元素的区别

1、伪类和伪元素的根本区别在于：它们**是否创造了新的元素**(抽象)。从我们模仿其意义的角度来看，如果需要添加新元素加以标识的，就是伪元素，反之，如果只需要在既有元素上添加类别的，就是伪类。

2、

伪类用一个冒号表示 :after

伪元素则使用两个冒号表示 ::after

3、

伪类就像真正的类一样，可以叠加使用，没有数量上限，只要不是互斥的

伪元素在一个选择器中只能出现一次，并且只能出现在末尾

41、数组和链表区别，分别适合什么数据结构

数组是将元素在内存中**连续存放**，由于**每个元素占用内存相同**，可以通过下标迅速访问数组中任何元素。但是如果要在数组中增加一个元素，需要移动大量元素，在内存中空出一个元素的空间，然后将要增加的元素放在其中。同样的道理，如果想删除一个元素，同样需要移动大量元素去填掉被移动的元素。如果应用需要快速访问数据，很少或不插入和删除元素，就应该用数组。

链表恰好相反，链表中的元素在内存中**不是顺序存储的，而是通过存在元素中的指针联系到一起**。比如：上一个元素有个指针指到下一个元素，以此类推，直到最后一个元素。如果要访问链表中一个元素，需要从第一个元素开始，一直找到需要的元素位置。但是增加和删除一个元素对于链表数据结构就非常简单了，只要修改元素中的指针就可以了。如果应用需要经常插入和删除元素你就需要用链表数据结构了

42、Array（数组）对象自带的方法

var arr = [0,1,2];

1.pop():删除数组的最后一个元素，减少数组的长度，返回删除的值。
这里是2.

2.push(3):将参数加载到数组的最后，返回新数组的长度。
现在arr中是：0,1,2,3

3.shift():删除数组的第一个元素,返回删除的值，同时数组长度减一。
这里是0

4.unshift(3,4):把参数加载数组的前面，返回新数组的长度。
现在arr:中是3,4,0,1,2

5.sort()：按指定的参数对数组进行排序，返回的值是经过排序之后的数组

6.reverse()：反转数组项的顺序，返回的值是经过排序之后的数组

7.concat(3,4)：把两个数组拼接起来。 返回的值是一个副本

8.slice(start,end)：返回从原数组中指定开始下标到结束下标之间的项组成的新数组，不会对原数组造成任何变化

9.splice()：会对原数组造成变化删除： 2个参数，起始位置，删除的项数插入： 3个参数，起始位置，删除的项数，插入的项替换：任意参数，起始位置，删除的项数，插入的任意数量的项

10.indexOf()和11.lastIndexOf()：接受两个参数，要查找的项(可选)和查找起点位置的索引
12.indexOf()：从数组开头向后查找
13.lastIndexOf()：从数组末尾开始向前查找

14.every()：对数组中的每一项运行给定函数，如果该函数对每一项都返回true，则返回true。
15.filter()：对数组中的每一项运行给定函数，返回该函数会返回true的项组成数组。
16.forEach()：对数组的每一项运行给定函数，这个方法没有返回值。
17.map()：对数组的每一项运行给定函数，返回每次函数调用的结果组成的数组。
18.some()：对数组的每一项运行给定参数，如果该函数对任一项返回true，则返回true。以上方法都不会修改数组中的包含的值。

19.reduce()和reduceRight()：缩小数组的方法，这两个方法都会迭代数组的所有项，然后构建一个最终返回的值。

20.join(separator)：将数组的元素组起一个字符串，以separator为分隔符，省略的话则用默认用逗号为分隔符

43、正则表达式匹配网址：

(https?|ftp|file)://[-A-Za-z0-9+&@#/%?=~_|!:,.;]+[-A-Za-z0-9+&@#/%=~_|]

IP 地址、前后有汉字、带参数的，都是 OK 的。

个人实习项目总结

1、使用到了什么技术
2、在实习中学到了什么。　　与 ui 人员的沟通交流，快速高效的还原 psd 设计图；和后台人员的沟通交流，学会了与后台进行数据交互。　学会了团队合作开发项目，管理好每个版本的代码。学会了如何构建一个前端网站。

3、项目中注重代码规范化的书写，

一、项目中的 requirejs

RequireJS 是一个非常小巧的 JavaScript 模块载入框架，是 AMD 规范最好的实现者之一
AMD 是"Asynchronous
Module Definition"的缩写，意思就是"异步模块定义"。它采用异步方式加载模块，模块的加载不影响它后面语句的运行。所有依赖这个模块的语句，都定义在一个回调函数中，等到加载完成之后，这个回调函数才会运行。还有一个特点是按需加载，已经加载过的就不会再加载了。
基本形式：
require([module], callback);

1、define([module] callback)定义模块

module:当前模块所以依赖的模块数组，数组的每个数组元素便是模块名或者叫模块 ID。（没有依赖就可以直接省略）
callback:模块的回调方法，用于保存模块具体的功能与代码，而这个回调函数又接收一个或者多个参数，这些参数会与模块数组的每个数组元素一一对应，即每个参数保存的是对应模块返回值。

```
define(["jquery", "../libs/load"], function ($, load) {
function pageTree(opts) {
//合并多个对象，把传入的 json 参数赋值到默认的 json 参数上，调用默认参数。
        this.opts = $.extend({}, pageTree.DEFAULTS, opts);
        var _ths = this.opts,
            _wrapID = $(_ths.wrapID);
}

    //默认参数
    pageTree.DEFAULTS = {
        url: '',
        data: '',
        md5: '',
        wrapID: '',
```

```
        title: '',
        todo:function(){}     //请求成功后的回调函数
    };

    return {     //返回定义的函数
        pageTree: pageTree
    };
});
```

2、require([module],callback)方法引入模块

//第一个参数是依赖的 js 文件数组，文件名就是模块 id；第二个参数就是回调函数，当所有依赖模块都加载完了就会执行这个回调函数。

```
require(["jquery", "libs/encryption", "module/pageTree", "module/deleteTree"], function ($, encryption, pageT, deleteT) {

})
```

3、html 页面中引入 require.js 和对应的页面 require()的 js 文件

```
<script type="text/javascript" src="../Js/libs/require.js" data-main="../Js/sysManage/OrgManage"></script>
```


人最大的无知就是不了解自己
人最大的问题就是觉得自己没有问题
羡慕别人比自己厉害
为什么不问自己
同样的时间你在做什么?
一个人是如何进步的
周而复始 最后获得进步
1、 模糊的问题自己的问题
2、 选对老师
3、 学习理论和经验， 改变思想
4、 实践
5、 反馈
6、 反思
7、 再实践
8、 再反馈
前端全栈交流学习圈：864305860
面向 1-3 年经验前端开发人员
帮助突破技术瓶颈， 提升思维能力