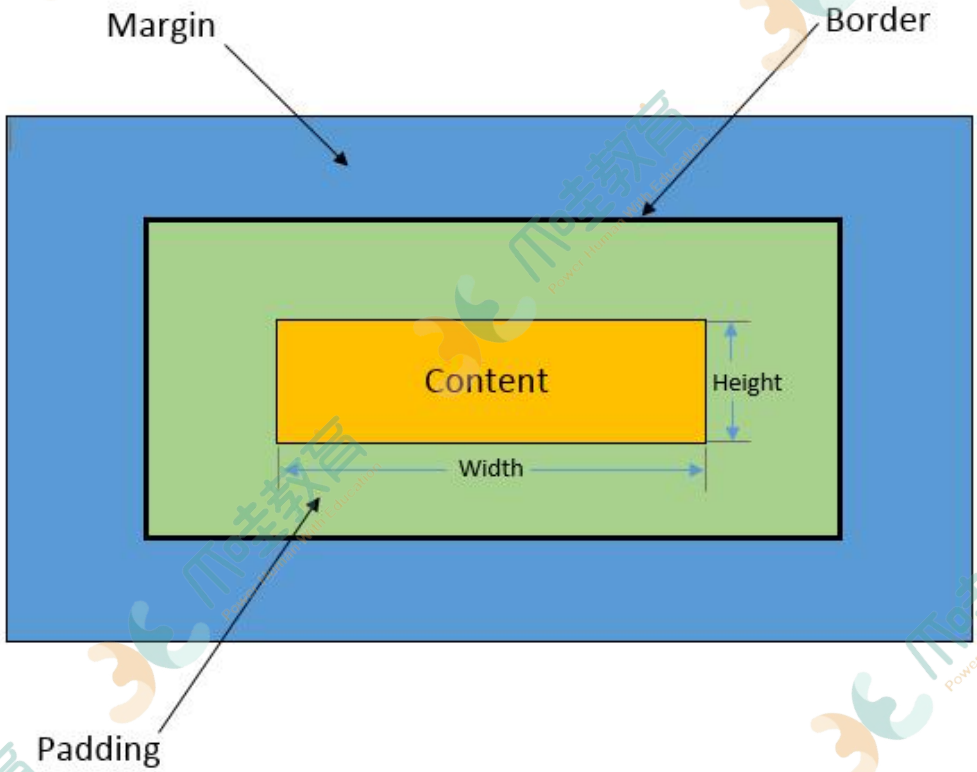
# 2022大厂前端面试题手册

# HTML、CSS、浏览器 相关：

## 1.网络中使用最多的图片格式有哪些

JPEG,GIF,PNG,最流行的是 jpeg 格式，可以把文件压缩到最小 在 ps 以 jpeg 格式存储时，提供 11 级压缩级别

## 2. 请简述 css 盒子模型




一个 css 盒子从外到内可以分成四个部分：margin（外边距），border（边框），padding（内边距），content（内容）。默认情况下，盒子的 width 和 height 属性只是设置 content（内容）的宽和高，

盒子真正的宽应该是：内容宽度+左右填充+左右边距+左右边框

盒子真正的高应该是：内容高度+上下填充+上下边距+上下边框

## 3.视频/音频标签的使用

视频：<video src="" ></video>
视频标签属性：

src 需要播放的视频地址
width/height 设置播放视频的宽高，和 img 标签的宽高属性一样
autoplay 是否自动播放
controls 是否显示控制条
poster 没有播放之前显示的展位图片
loop 是否循环播放
perload 预加载视频（缓存）与 autoplay 相冲突，设置了 autoplay 属性，perload 属性会失效。
muted 静音模式
音频: 音频属性和视频属性差不多，不过宽高和 poster 属性不能用
<audio>
<source src="" type="" >
</audio>

## 4.HTML5 新增的内容有哪些

新增语义化标签

新增表单类型
表单元素
表单属性
表单事件
多媒体标签

## 5.Html5 新增的语义化标签有哪些

语义化标签优点：1.提升可访问性 2.seo 3.结构清晰，利于维护
Header 页面头部 main 页面主要内容 footer 页面底部
Nav 导航栏 aside 侧边栏 article 加载页面一块独立内容
Section 相当于 div figure 加载独立内容（上图下字）
figcaption figure 的标题
Hgroup 标题组合标签 mark 高亮显示 dialog 加载对话框标签（必须配合 open 属性）
Embed 加载插件的标签 video 加载视频 audio 加载音频（支持格式 ogg，mp3，wav）

## 6.Css3 新增的特性

边框：
border-radios 添加圆角边框
border-shadow：给框添加阴影 （水平位移，垂直位移，模糊半径，阴影尺寸，阴影颜色，insetr（内/外部阴影））
border-image：设置边框图像
border-image-source 边框图片的路径

border-image-slice 图片边框向内偏移
border-image-width 图片边框的宽度
border-image-outset 边框图像区域超出边框的量
border-image-repeat 图像边框是否平铺（repeat 平铺 round 铺满 stretch 拉伸）
背景：
Background-size 背景图片尺寸
Background-origin 规定 background-position 属性相对于什么位置定位
Background-clip 规定背景的绘制区域（padding-box，border-box，content-box）
渐变：
Linear-gradient（）线性渐变
Radial-gradient（）径向渐变
文本效果：
Word-break：定义如何换行
Word-wrap：允许长的内容可以自动换行
Text-overflow：指定当文本溢出包含它的元素，应该干啥
Text-shadow：文字阴影（水平位移，垂直位移，模糊半径，阴影颜色）
转换：
Transform 应用于 2D3D 转换，可以将元素旋转，缩放，移动，倾斜
Transform-origin 可以更改元素转换的位置，（改变 xyz 轴）
Transform-style 指定嵌套元素怎么样在三位空间中呈现
2D 转换方法：
rotate 旋转 translate（x，y）指定元素在二维空间的位移 scale（n）定义缩放转换
3D 转换方法：
Perspective（n）为 3D 转换 translate rotate scale

过渡：
Transition-proprety 过渡属性名
Transition-duration 完成过渡效果需要花费的时间
Transition-timing-function 指定切换效果的速度
Transition-delay 指定什么时候开始切换效果
动画：animation
Animation-name 为@keyframes 动画名称
animation-duration 动画需要花费的时间
animation-timing-function 动画如何完成一个周期
animation-delay 动画启动前的延迟间隔
animation-iteration-count 动画播放次数
animation-direction 是否轮流反向播放动画

**7.清除浮动的方式有哪些？请说出各自的优点**

高度塌陷：当所有的子元素浮动的时候，且父元素没有设置高度，这时候父元素就会产生高度塌陷。
清除浮动方式 1：给父元素单独定义高度
优点：快速简单，代码少 缺点：无法进行响应式布局
清除浮动方式 2：父级定义 overflow：hidden；zoom：1（针对 ie6 的兼容）
优点：简单快速、代码少，兼容性较高 缺点：超出部分被隐藏，布局时要注意
清除浮动方式 3：在浮动元素后面加一个空标签，clear：both；height：0；overflow：hidden
优点：简单快速、代码少，兼容性较高。
缺点：增加空标签，不利于页面优化
清除浮动方式 4：父级定义 overflow：auto

优点：简单，代码少，兼容性好

缺点：内部宽高超过父级 div 时，会出现滚动条

清除浮动方式 5：万能清除法：

给塌陷的元素添加伪对象

```
.father：after{
        Content：“随便写”；
        Clear：both；
        display：block；
        Height：0；
        Overflow：hidden；
        Visibility：hidden
}
```

优点：写法固定，兼容性高　　缺点：代码多

**8.定位的属性值有何区别**

Position 有四个属性值：relative absolute fixed static

Relative 相对定位 不脱离文档流，相对于自身定位

Absolute 绝对定位，脱离文档流 相对于父级定位

Fixed 固定定位，脱离文档流，相对于浏览器窗口定位

Static 默认值，元素出现在正常的流中

**9.子元素如何在父元素中居中**

水平居中：

1.子父元素宽度固定，子元素设置 margin：auto，并且子元素不能设置浮动，否则居中失效
2.子父元素宽度固定，父元素设置 text-align：center，子元素设置 display：inline-block，并且子元素不能设置浮动，否则居中失效
水平垂直居中：
子元素相对于父元素绝对定位，子元素 top，left 设置 50%，子元素 margin-top 和 margin-left 减去各自宽高的一半
子元素相对于父元素绝对定位，子元素上下左右全为 0，然后设置子元素 margin：auto
父元素设置 display：table-cell vertical-align：middle，子元素设置 margin：auto
子元素相对定位，子元素 top，left 值为 50%，transform：translate（-50%，-50%）
子元素相对父元素绝对定位，子元素 top，left 值为 50%，transform：translate（-50%，-50%）
父元素设置弹性盒子，
display：flex；　justfy-content：center　；align-item：center；
justfy-content：center

**10.Border-box 与 content-box 的区别**

Content-box 标准盒模型 width 不包括 padding 和 border
Border-box 怪异盒模型 width 包括 padding 和 border

**11.元素垂直居中**

1.设置子元素和父元素的行高一样
2.子元素设置为行内块，再加 vertical-align：middle
3.已知父元素高度,子元素相对定位,通过 transform:translateY(-50%)
4.不知道父元素高度，子绝父相，子元素 top：50%，transform：translateY（-50%)
5.创建一个隐藏节点，让隐藏节点的 height 为剩余高度的一半
6.给父元素 display：table，子元素 display：table-cell，vertical-align：middle
7.给父元素添加伪元素
8.弹性盒，父元素 display：flex，子元素 align-self：center

**12.如何让 chrome 浏览器显示小于 12px 的文字**

本来添加谷歌私有属性 -webkit-text-size-adjust： none，现在 -webkit-transform：scale（）

**13.Css 选择器有哪些，那些属性可以继承，优先级如何计算？**

**Css3 新增的伪类有哪些**

Css2 选择器：
元素选择器，id 选择器，群组选择器，类选择器，*通配符选择器，后代选择器
Css2 伪类选择器：a:link/visited/hover/active
Css3 选择器：

空格 > +相邻兄弟选择器 ~通用选择器（查找后面所有）
结构伪类选择器：
查找第几个 nth-child（n）
查找同一类型第几个 nth-of-type
查找唯一类型 only-of-type
属性选择器：根据标签属性查找 [attr=value]
: root 查找根元素 html 标签
: empty 查赵空标签
目标伪类选择器：（表单）
： enabled 查找可以使用的标签
： disabled 查找禁止使用的标签
： checked 查找被选中的标签
伪元素选择器 ：：selection 设置选中文本内容的高亮显示（只能用于背景色和文本颜色）
否定伪类选择器 not（）
语言伪类选择器 lang（取值）
优先级（权重）：
元素选择器 1
伪元素选择器 1
class 选择器 10
伪类选择器 10
属性选择器 10
Id 选择器 100
内联样式的权重 1000
包含选择器权重为权重之和
继承样式权重为 0
那些属性可以继承：
Css 继承特性主要是文本方面

所有元素可继承：visibility 和 cursor
块级元素可继承：text-indent 和 text-align
列表元素可继承：list-style，list-style-type，list-style-position，list-style-image
内联元素可继承：letter-spacing，word-spacing，line-height，color，font，font-family，font-size
Font-style，font-variant，font-weight，text-decoration，text-transform，direction
字母间距 段落间距 行高 字体颜色 字体种类 字体大小 字体样式 字体粗细 小型大写字母文本 文本修饰 转换不同元素中的文本 文本方向

**14.网页中有大量图片加载很慢 你有什么办法进行优化？**

1.图片懒加载，在图片未可视区域加一个滚动条事件，判断图片位置与浏览器顶端和页面的距离，如果前者小鱼后者，优先加载
2.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
3.使用 csssprite 或者 svgsprite

**15.行内元素/块级元素有哪些？**

行内元素：相邻的行内元素会排列在同一行，不会独占一行 设置宽高无效 span
块级元素：会独占一行 可以设置宽高等属性 div
可变元素：根据上下文预警决定该元素为块元素还是内联元素

块级元素：div h1-h6 hr p ul ol table address blockquote dir from menu
行内元素：a br I em img input select span sub sup u textarea
可变元素：button del iframe ins

## 16.浏览器的标准模式和怪异模式区别？

标准模式：浏览器按照 W3C 标准解析执行代码
怪异模式：浏览器根据自己的方式解析执行代码，因为不同浏览器解析执行方式不一样，所以叫怪异模式
区别：
在怪异模式下，盒模型为怪异盒模型 而在标准模式下为标准盒子模型
图片元素的垂直对齐方式对于行内元素和 table-cell 元素，标准模式下 vertical-align 属性默认值是 baseline，而在怪异模式下，table 单元格中的图片的 vertical-align 属性默认值是 bottom，因此在图片底部会有几像素的空间
元素中的字体 css 中 font 的属性都是可以继承的，怪异模式下，对于 table 元素，字体的某些元素不能从其他封装元素继承中得到，特别是 font-size 属性
内联元素的尺寸标准模式下，non-replaced inline 元素无法自定义大写，怪异模式下，定义元素的宽高会影响元素的尺寸
元素的百分比高度当一个元素使用百分比高度时，在标准模式下，高度取决于内容变化，在怪异迷失下，百分比被准确应用
元素溢出的处理标准模式下，overflow 取值默认值为 visible，在怪异模式下，这个溢出会被当做扩展 box 对待，就是元素的大小由内容决定，溢出不会裁剪，元素框自动调整，包含溢出内容

## 17.Margin 和 padding 在什么场合下使用

Margin 外边距 自身边框到另一个边框之间的距离
Padding 内边距 自身边距到自身内容之间的距离
当需要在 border 外侧添加空白时用 margin，当需要在 border 内侧添加空白时用 padding

## 18.弹性盒子布局属性有那些请简述?

Flex-direction：弹性容器中子元素排列方式（主轴排列方式）
Flex-wrap：设置弹性盒子的子元素超出父容器时是否换行
Flex-flow：是 flex-direction 和 flex-wrap 简写形式
Align-item：设置弹性盒子元素在侧轴上的对齐方式
Align-content：设置行对齐
Justify-content：设置弹性盒子元素在主轴上的对齐方式

## 19.怎么实现标签的禁用

添加 disabled 属性

## 20.Flex 布局原理

就是通过给父盒子添加 flex 属性，来控制子盒子的位置和排列方式

## 21.Px，rem，em 的区别

Px，绝对长度单位，像素 px 是相对于显示器屏幕分辨率来说的

em 相对长度单位，相对于当前对象内文本的字体尺寸

em 的值并不是固定的

em 会继承父级元素的字体大小（参考物是父元素的 font-size）

em 中所有的字体都是相对于父元素的大小决定的

rem 相对于 html 根元素的 font-size

1em=1rem=16px 在 body 中加入 font-size：62.5% 这样直接就是原来的 px 数值除以 10 加上 em 就可以

## 22.网页的三层结构有哪些

结构（html 或 xhtm 标记语言）表现（css 样式表）行为（js）

请简述媒体查询

媒体查询扩展了 media 属性，就是根据不同的媒体类型设置不同的 css 样式，达到自适应的目的。

Rem 缺点

比如：小说网站，屏幕越小的移动设备如果用了 rem 肯定文字就越小，就会导致看文章的时候特别费眼

## 25.常见的兼容性一阶段内容中记几个

## 26.垂直与水平居中的方式

## 27.三栏布局方式两边固定中间自适应

1. margin 负值法：左右两栏均左浮动，左右两栏采用负的 margin 值。中间栏被宽度为 100%的浮动元素包起来

2. 自身浮动法：左栏左浮动，右栏右浮动，中间栏放最后

3. 绝对定位法：左右两栏采用绝对定位，分别固定于页面的左右两侧，中间的主体栏用左右 margin 值撑开距离。

4.flex 左右固定宽 中间 flex：1

5.网格布局

6. table 布局

## 28.Doctype 作用

声明文档类型

## 29.说一下 HTML5 drag api

参考回答:

dragstart：事件主体是被拖放元素，在开始拖放被拖放元素时触发，。
darg：事件主体是被拖放元素，在正在拖放被拖放元素时触发。
dragenter：事件主体是目标元素，在被拖放元素进入某元素时触发。
dragover：事件主体是目标元素，在被拖放在某元素内移动时触发。
dragleave：事件主体是目标元素，在被拖放元素移出目标元素是触发。
drop：事件主体是目标元素，在目标元素完全接受被拖放元素时触发。
dragend：事件主体是被拖放元素，在整个拖放操作结束时触发

## 30 对 HTML 语义化标签的理解

HTML5 语义化标签是指正确的标签包含了正确的内容，结构良好，便于阅读,比如 nav 表示导航条,类似的还有 article、header、footer 等等标签。

## 31.web 性能优化

降低请求量：合并资源，减少 HTTP 请求数，minify / gzip 压缩，webP，lazyLoad。加快请求速度：预解析 DNS，减少域名数，并行加载，CDN 分发。
缓存：HTTP 协议缓存请求，离线缓存 manifest，离线数据缓存 localStorage。渲染:JS/CSS 优化,加载顺序,服务端渲染,pipeline。

## 32.浏览器缓存机制

缓存分为两种：强缓存和协商缓存，根据响应的 header 内容来决

强缓存相关字段有 expires，cache-control。如果 cache-control 与 expires 同时存在的话， cache-control 的优先级高于 expires。

协商缓存相关字段有 Last-Modified/If-Modified-Since，Etag/If-None-Match

## 33.浏览器输入网址到页面渲染全过程

DN

解析 TCP

连接

发送 HTTP 请求

服务器处理请求并返回 HTTP 报文浏览器解析渲染页面

连接结束

## 34.画一条 0.5px 的线

采用 meta viewport 的方式

```
<meta
name="viewport"
content="initial-scale=1.
0,  maximum-scale=1.0,
user-scalable=no" />
```

采用 border-image 的方式 采用 transform: scale() 的方式

## 35.关于 JS 动画和 css3 动画的差异性

渲染线程分为 main thread 和 compositor thread，如果 css 动画只改变 transform 和 opacity，这时整个 CSS 动画得以在 compositor trhead 完成（而 JS 动画则会在 main thread 执行，然后出发 compositor thread 进行下一步操作），特别注意的是如果改变 transform 和 opacity 是不会 layout 或者 paint 的。
区别：
功能涵盖面，JS 比 CSS 大
实现/重构难度不一，CSS3 比 JS 更加简单，性能跳优方向固定对帧速表现不好的低版本浏览器，css3 可以做到自然降级
css 动画有天然事件支持 css3 有兼容性问题

## 36.双边距重叠问题（外边距折叠）

多个相邻（兄弟或者父子关系）普通流的块元素垂直方向 marigin 会重叠折叠的结果为：
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。

## 37.浮动清除

方法一：使用带 clear 属性的空元素在浮动元素后使用一个空元素如 ，并在 CSS 中赋予.clear{clear:both;}属性即可清理浮动。亦可使用<br class="clear" />或<hr class="clear" />来进行清理。

方法二：使用 CSS 的 overflow 属性

给浮动元素的容器添加 overflow:hidden;或 overflow:auto;可以清除浮动，另外在 IE6 中还需要触发 hasLayout ，例如为父元素设置容器宽高或设置 zoom:1。

在添加 overflow 属性后，浮动元素又回到了容器层，把容器高度撑起，达到了清理浮动的效果。

方法三：给浮动的元素的容器添加浮动

给浮动元素的容器也添加上浮动属性即可清除内部浮动，但是这样会使其整体浮动，影响布局，不推荐使用。

方法四：使用邻接元素处理

什么都不做，给浮动元素后面的元素添加 clear 属性。方法五：使用 CSS 的:after 伪元素

结合:after 伪元素（注意这不是伪类，而是伪元素，代表一个元素之后最近的元素）和 IEhack ，可以完美兼容当前主流的各大浏览器，这里的 IEhack 指的是触发 hasLayout。给浮动元素的容器添加一个 clearfix 的 class，然后给这个 class 添加一个:after 伪元素实现元素末尾添加一个看不见的块元素（Block element）清理浮动。

## 38.CSS 选择器有哪些，优先级呢

id 选择器，class 选择器，标签选择器，伪元素选择器，伪类选择器等同一元素引用了多个样式时，排在后面的样式属性的优先级高；

样式选择器的类型不同时，优先级顺序为：id 选择器 > class 选择器 > 标签选择器； 标签之间存在层级包含关系时，后代元素会继承祖先元素的样式。如果后代元素定义了与祖先元素相同的样式，则祖先元素的相同的样式属性会被覆盖。继承的样式的优先级比较低，至少比标签选择器的优先级低；

带有!important 标记的样式属性的优先级最高；

样式表的来源不同时，优先级顺序为：内联样式> 内部样式 > 外部样式 > 浏览器用户自定义样式 > 浏览器默认样式

## 39.css 动画如何实现

创建动画序列，需要使用 animation 属性或其子属性，该属性允许配置动画时间、时长以及其他动画细节，但该属性不能配置动画的实际表现，动画的实际表现是由@keyframes 规则实现，具体情况参见使用 keyframes 定义动画序列小节部分。 transition 也可实现动画。transition 强调过渡，是元素的一个或多个属性发生变化时产生的过渡效果，同一个元素通过两个不同的途径获取样式，而第二个途径当某种改变发生

（例如 hover）时才能获取样式，这样就会产生过渡动画。

## 40.如何实现元素的垂直居中

法一：父元素 display:flex,align-items:center;

法二：元素绝对定位，top:50%，margin-top：

-（高度/2）法三：高度不确定用 transform：

translateY（-50%）

法四：父元素 table 布局，子元素设置 vertical-align:center;

## 41.CSS3 中对溢出的处理

cnkOhu

text-overflow 属性，值为 clip 是修剪文本；ellipsis 为显示省略符号来表被修剪的文本； string 为使用给定的字符串来代表被修剪的文本。

## 42.对 CSS 的新属性有了解过的吗？

CSS3 的新特性中，在布局方面新增了 flex 布局，在选择器方面新增了例如

first-of-type,nth-child 等选择器，在盒模型方面添加了 box-sizing 来改变盒模型，在动画方面增加了 animation，2d 变换，3d 变换等，在颜色方面添加透明，rbga 等，在字体方面允许嵌入字体和设置字体阴影，最后还有媒体查讯等

## 43.overflow 的原理

要讲清楚这个解决方案的原理，首先需要了解块格式化上下文，A block formatting context is a part of a visual CSS rendering of a Web page. It is the region in which the layout of block boxes occurs and in which floats interact with each other.翻译过来就

是块格式化上下文是 CSS 可视化渲染的一部分，它是一块区域，规定了内部块盒 的渲染方式，以及浮动相互之间的影响关系
当元素设置了 overflow 样式且值部位 visible 时，该元素就构建了一个 BFC，BFC 在计算高度时，内部浮动元素的高度也要计算在内，也就是说技术 BFC 区域内只有一个浮动元素，BFC 的高度也不会发生塌缩，所以达到了清除浮动的目的。

## 44.css 定位

固定定位 fixed：
元素的位置相对于浏览器窗口是固定位置，即使窗口是滚动的它也不会移动。Fixed 定位使元素的位置与文档流无关，因此不占据空间。Fixed 定位的元素和其他元素重叠。相对定位 relative：
如果对一个元素进行相对定位，它将出现在它所在的位置上。然后，可以通过设置垂直或水平位置，让这个元素“相对于”它的起点进行移动。 在使用相对定位时，无论是否进行移动，元素仍然占据原来的空间。因此，移动元素会导致它覆盖其它框。
绝对定位 absolute：
绝对定位的元素的位置相对于最近的已定位父元素，如果元素没有已定位的父元素，那么它的位置相对于<html>。absolute 定位使元素的位置与文档流无关，因此不占据空间。 absolute 定位的元素和其他元素重叠。
粘性定位 sticky：
元素先按照普通文档流定位，然后相对于该元素在流中的 flow root（BFC）和 containing block（最近的块级祖先元素）定位。而后，元素定位表现为在跨越特定阈值前为相对定 位，之后为固定定位。
默认定位 Static：

默认值。没有定位，元素出现在正常的流中（忽略 top, bottom, left, right 或者 z-index 声明）。
inherit:
规定应该从父元素继承 position 属性的值。

# Javascript 相关

## 1.Js 基本数据类型有哪些

字符串 String 数值 Number 布尔 boolean null undefined 对象 数组

## 2.Ajax 如何使用

一个完整的 AJAX 请求包括五个步骤：

创建 XMLHTTPRequest 对象
使用 open 方法创建 http 请求，并设置请求地址
xhr.open（get/post，url，async，true（异步），false（同步））经常使用前三个参数
设置发送的数据，用 send 发送请求
注册事件（给 ajax 设置事件）
获取响应并更新页面

## 3.如何判断一个数据是 NaN

NaN 非数字 但是用 typeof 检测是 number 类型
利用 NaN 的定义 用 typeof 判断是否为 number 类型并且判断是否满足 isnan
利用 NaN 是唯一一个不等于任何自身的特点 n！==n
利用 ES6 中提供的 Object.is()方法（判断两个值是否相等） n==nan
Js 中 null 与 undefined 区别
相同点：用 if 判断时，两者都会被转换成 false
不同点：
number 转换的值不同 number（null）为 0 number（undefined）为 NaN
Null 表示一个值被定义了，但是这个值是空值
Undefined 变量声明但未赋值

## 5.闭包是什么？有什么特性？对页面会有什么影响

闭包可以简单理解成：定义在一个函数内部的函数。其中一个内部函数在包含它们的外部函数之外被调用时，就会形成闭包。
特点：
1.函数嵌套函数。
2.函数内部可以引用外部的参数和变量。
3.参数和变量不会被垃圾回收机制回收。
使用：
1.读取函数内部的变量；
2.这些变量的值始终保持在内存中，不会在外层函数调用后被自动清除。

优点：
1:变量长期驻扎在内存中；
2:避免全局变量的污染；
3:私有成员的存在 ；
缺点：会造成内存泄露

**6. Js 中常见的内存泄漏：**

1.意外的全局变量
2.被遗忘的计时器或回调函数
3.脱离 DOM 的引用
4.闭包

**7.事件委托是什么？如何确定事件源（Event.target 谁调用谁就是事件源）**

JS 高程上讲：事件委托就是利用事件冒泡，只制定一个时间处理程序，就可以管理某一类型的所有事件。
事件委托，称事件代理，是 js 中很常用的绑定事件的技巧，事件委托就是把原本需要绑定在子元素的响应事件委托给父元素，让父元素担当事件监听的职务，事件委托的原理是 DOM 元素的事件冒泡

## 8.什么是事件冒泡？

一个事件触发后，会在子元素和父元素之间传播，这种传播分为三个阶段，
捕获阶段（从 window 对象传导到目标节点（从外到里），这个阶段不会响应任何事件），目标阶段，（在目标节点上触发），冒泡阶段（从目标节点传导回 window 对象（从里到外）），事件委托/事件代理就是利用事件冒泡的机制把里层需要响应的事件绑定到外层

## 9.本地存储与 cookie 的区别

Cookie 是小甜饼的意思。顾名思义，cookie 确实非常小，它的大小限制为 4KB 左右。它的主要用途有保存登录信息，比如你登录某个网站市场可以看到“记住密码”，这通常就是通过在 Cookie 中存入一段辨别用户身份的数据来实现的。

localStorage

localStorage 是 HTML5 标准中新加入的技术，它并不是什么划时代的新东西。早在 IE 6 时代，就有一个叫 userData 的东西用于本地存储，而当时考虑到浏览器兼容性，更通用的方案是使用 Flash。而如今，localStorage 被大多数浏览器所支持，如果你的网站需要支持 IE6+，那以 userData 作为你方案是种不错的选择。

sessionStorage

sessionStorage 与 localStorage 的接口类似，但保存数据的生命周期与 localStorage 不同。做过后端开发的同学应该知道 Session 这个词的意思，直译过来是“会话”。而 sessionStorage 是一个前端的概念，它只是可以将一部分数据在当前会话中保存下来，刷新页面数据依旧存在。但当页面关闭后，sessionStorage 中的数据就会被清空。

三者的异同

特性

Cookie

localStorage

sessionStorage

数据的生命期

一般由服务器生成，可设置失效时间。如果在浏览器端生成 Cookie，默认是关闭浏览器后失效

除非被清除，否则永久保存

仅在当前会话下有效，关闭页面或浏览器后被清除

存放数据大小

4K 左右

一般为 5MB

与服务器端通信

每次都会携带在 HTTP 头中，如果使用 cookie 保存过多数据会带来性能问题

仅在客户端（即浏览器）中保存，不参与和服务器的通信

易用性

需要程序员自己封装，源生的 Cookie 接口不友好

源生接口可以接受，亦可再次封装来对 Object 和 Array 有更好的支持

## 10.ES6 新特性

const 和 let、模板字符串、箭头函数、函数的参数默认值、对象和数组解构、for...of 和 for...in、ES6 中的类

## 11.Let 与 var 与 const 的区别

Var 声明的变量会挂载在 window 上，而 let 和 const 声明的变量不会
Var 声明的变量存在变量提升，let 和 const 不存在变量提升
同一作用域下 var 可以声明同名变量，let 和 const、不可以
Let 和 const 声明会形成块级作用域
Let 暂存死区
Const 一旦声明必须赋值，不能用 null 占位，声明后不能再修改，如果声明的是复合类型数据，可以修改属性

## 12.数组方法有哪些请简述

push() 从后面添加元素，返回值为添加完后的数组的长度
arr.pop() 从后面删除元素，只能是一个，返回值是删除的元素
arr.shift() 从前面删除元素，只能删除一个 返回值是删除的元素
arr.unshift() 从前面添加元素，返回值是添加完后的数组的长度
arr.splice(i,n) 删除从 i(索引值)开始之后的那个元素。返回值是删除的元素
arr.concat() 连接两个数组 返回值为连接后的新数组

str.split() 将字符串转化为数组

arr.sort() 将数组进行排序,返回值是排好的数组,默认是按照最左边的数字进行排序，不是按照数字大小排序的

arr.reverse() 将数组反转,返回值是反转后的数组

arr.slice(start,end) 切去索引值 start 到索引值 end 的数组,不包含 end 索引的值，返回值是切出来的数组

arr.forEach(callback) 遍历数组,无 return 即使有 return，也不会返回任何值，并且会影响原来的数组

arr.map(callback) 映射数组(遍历数组),有 return 返回一个新数组 。

arr.filter(callback) 过滤数组，返回一个满足要求的数组

**13.Json 如何新增/删除键值对**

**14.什么是面向对象请简述**

面向对象是一种思想，是基于面向过程而言的，就是说面向对象是将功能等通过对象来实现，将功能封装进对象之中，让对象去实现具体的细节；这种思想是将数据作为第一位，这是对数据一种优化，操作起来更加的方便，简化了过程。

Js 本身是没有 class 类型的，但是每个函数都有一个 prototype 属性，prototype 指向一个对象，当函数作为构造函数时，prototype 就起到类似于 class 的作用

面向对象有三个特点：封装（隐藏对象的属性和实现细节，对外提供公共访问方式），

继承（提高代码复用性，继承是多态的前提），多态（是父类或接口定义

的引用变量可以指向子类或具体实现类的实例对象）

## 15.普通函数和构造函数的区别

1.构造函数也是一个普通函数，创建方式和普通函数一样，但是构造函数习惯上首字母大写
2.调用方式不一样，普通函数直接调用，构造函数要用关键字 new 来调用
3.调用时，构造函数内部会创建一个新对象，就是实例，普通函数不会创建新对象
4.构造函数内部的 this 指向实例，普通函数内部的 this 指向调用函数的对象（如果没有对象调用，默认为 window）
5.构造函数默认的返回值是创建的对象（也就是实例），普通函数的返回值由 return 语句决定
6.构造函数的函数名与类名相同

## 16.请简述原型/原型链/（原型）继承

什么是原型：

任何对象实例都有一个原型，也叫原型对象，这个原型对象由对象的内置属性_proto_指向它的构造函数的 prototype 指向的对象，即任何对象都是由一个构造函数创建的，但是不是每一个对象都有 prototype，只有方法才有 prototype。

什么是原型链?

原型链基本思想是利用原型让一个引用类型继承另一个引用类型的属性

和方法。我们知道，每个构造函数都有一个原型对象，每个原型对象都有一个指向构造函数的指针，而实例又包涵一个指向原型对象的内部指针。

原型链的核心就是依赖对象的_proto_的指向，当自身不存在的属性时，就一层层的扒出创建对象的构造函数，直至到 Object 时，就没有_proto_指向了。

因为_proto_实质找的是 prototype，所以我们只要找这个链条上的构造函数的 prototype。其中 Object.prototype 是没有_proto_属性的，它==null。

每个构造函数都有一个原型对象，原型对象都包含一个指向构造函数的指针，而实例都包含指向原型对象内部的指针。我们让原型对象（1）等于另一个原型对象的实例（2），

此时原型对象（2）将包含一个指向原型对象（1）的指针，

再让原型对象（2）的实例等于原型对象（3），如此层层递进就构成了实例和原型的链条，这就是原型链的概念

每个构造函数都有一个原型对象，原型对象都包含一个指向构造函数想指针(constructor)，而实例对象都包含一个指向原型对象的内部指针(__proto__)。如果让原型对象等于另一个原型对象的实例，此时的原型对象将包含一个指向另一个原型的指针(__proto__)，另一个原型也包含着一个指向另一个构造函数的指针(constructor)。假如另一个原型又是另一个类型的实例……这就构成了实例与原型的链条。也叫原型链

原型继承是 js 的一种继承方式，原型链作为实现继承的主要方法,其基本思路是利用原型让一个引用类型继承另一个引用类型的属性和方法,

原型继承：利用原型中的成员可以被和其相关的对象共享这一特性，可以实现继承，这种实现继承的方式，就叫做原型继承.

## 17.Promise 的理解

一、什么是 Promise?

我们都知道，Promise 是承诺的意思，承诺它过一段时间会给你一个结果。

Promise 是一种解决异步编程的方案，相比回调函数和事件更合理和更强大。

从语法上讲，promise 是一个对象，从它可以获取异步操作的消息；

二、promise 有三种状态：pending 初始状态也叫等待状态，fulfiled 成功状态，rejected 失败状态；状态一旦改变，就不会再变。创造 promise 实例后，它会立即执行。

三、Promise 的两个特点

1、Promise 对象的状态不受外界影响

2、Promise 的状态一旦改变，就不会再变，任何时候都可以得到这个结果，状态不可以逆，

四、Promise 的三个缺点

1）无法取消 Promise,一旦新建它就会立即执行，无法中途取消

2）如果不设置回调函数，Promise 内部抛出的错误，不会反映到外部

3）当处于 pending（等待）状态时，无法得知目前进展到哪一个阶段，是刚刚开始还是即将完成

## 18.我们用 Promise 来解决什么问题？

promise 是用来解决两个问题的：

1.回调地狱，代码难以维护， 常常第一个的函数的输出是第二个函数的输入这种现象

2.promise 可以支持多并发的请求，获取并发请求中的数据

这个 promise 可以解决异步的问题，本身不能说 promise 是异步的

**19.请简述 async 的用法**

Async 就是 generation 和 promise 的语法糖，async 就是将 generator 的*换成 async，将 yiled 换成 await

函数前必须加一个 async，异步操作方法前加一个 await 关键字，意思就是等一下，执行完了再继续走，注意：await 只能在 async 函数中运行，否则会报错

Promise 如果返回的是一个错误的结果，如果没有做异常处理，就会报错，所以用 try..catch 捕获一下异常就可以了

**20.. 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，然后服务器响应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

3. 此时，Web 服务器提供资源服务，客户端开始下载资源。

## 21.get 请求传参长度的误区

误区：我们经常说 get 请求参数的大小存在限制，而 post 请求的参数大小是无限制的。实际上 HTTP 协议从未规定 GET/POST 的请求长度限制是多少。对 get 请求参数的限制是来源与浏览器或 web 服务器，浏览器或 web 服务器限制了 url 的长度。为了明确这个概念，我们必须再次强调下面几点:
HTTP 协议 未规定 GET 和 POST 的长度限制
GET 的最大长度显示是因为 浏览器和 web 服务器限制了 URI 的长度不同的浏览器和 WEB 服务器，限制的最大长度不一样要支持 IE，则最大长度为 2083byte，若只支持 Chrome，则最大长度 8182byte

## 22.补充 get 和 post 请求在缓存方面的区别

post/get 的请求区别，具体不再赘述。
补充补充一个 get 和 post 在缓存方面的区别：get 请求类似于查

找的过程，用户获取数据，可以不用每次都与数据库连接，所以可以使用缓存。

post 不同，post 做的一般是修改和删除的工作，所以必须与数据库交互，所以不能使用缓存。因此 get 请求适合于请求缓存。

## 23.说一下闭包

一句话可以概括：闭包就是能够读取其他函数内部变量的函数，或者子函数在外调用， 子函数所在的父函数的作用域不会被释放。

## 24.说说前端中的事件流

HTML 中与 javascript 交互是通过事件驱动来实现的，例如鼠标点击事件 onclick、页面的滚动事件 onscroll 等等，可以向文档或者文档中的元素添加事件侦听器来预订事件。想要知道这些事件是在什么时候进行调用的，就需要了解一下“事件流”的概念。

什么是事件流：事件流描述的是从页面中接收事件的顺序,DOM2 级事件流包括下面几个阶段。

事件捕获阶段处于目标阶段事件冒泡阶段

addEventListener：addEventListener 是 DOM2 级事件新增的指定事件处理程序的操作，这个方法接收 3 个参数：要处理的事件名、作为事件处理程序的函数和一个布尔值。最后这个布尔值参数如果是 true，表示在捕获阶段调用事件处理程序；如果是 false，表示 在冒泡阶段调用事件处理程序。

IE 只支持事件冒泡。

## 25.说一下事件委托

简介：事件委托指的是，不在事件的发生地（直接 dom）上设置监听函数，而是在其父元素上设置监听函数，通过事件冒泡，父元素可以监听到子元素上事件的触发，通过判断事件发生元素 DOM 的类型，来做出不同的响应。

举例：最经典的就是 ul 和 li 标签的事件监听，比如我们在添加事件时候，采用事件委托机制，不会在 li 标签上直接添加，而是在 ul 父元素上添加。

好处：比较合适动态元素的绑定，新添加的子元素也会有监听函数，也可以有事件触发机制。

## 26.JS 的 new 操作符做了哪些事情

new 操作符新建了一个空对象，这个对象原型指向构造函数的 prototype，执行构造函数后返回这个对象。

## 27.改变函数内部 this 指针的指向函数（bind，apply，call 的区别）

通过 apply 和 call 改变函数的 this 指向，他们两个函数的第一个参数都是一样的表示要改变指向的那个对象，第二个参数，apply 是数组，而 call 则是 arg1,arg2...这种形式。通过 bind 改变 this 作用域会返回一个新的函数，这个函数不会马上执行。

## 28.JS 的各种位置，比 clientHeight,scrollHeight,offsetHeight ,以及 scrollTop, offsetTop,clientTop 的区别？

clientHeight：表示的是可视区域的高度，不包含 border 和滚动条 offsetHeight：表示可视区域的高度，包含了 border 和滚动条 scrollHeight：表示了所有区域的高度，包含了因为滚动被隐藏的部分。clientTop：表示边框 border 的厚度，在未指定的情况下一般为 0scrollTop：滚动后被隐藏的高度，获取对象相对于由 offsetParent 属性指定的父坐标(css 定位的元素或 body 元素)距离顶端的高度。

## 29.JS 拖拽功能的实现

首先是三个事件，分别是 mousedown，mousemove，mouseup 当鼠标点击按下的时候，需要一个 tag 标识此时已经按下，可以执行 mousemove 里面的具体方法。clientX，clientY 标识的是鼠标的坐标，分别标识横坐标和纵坐标，并且我们用 offsetX 和 offsetY 来表示元素的元素的初始坐标，移动的举例应该是：鼠标移动时候的坐标-鼠标按下去时候的坐标。也就是说定位信息为：鼠标移动时候的坐标-鼠标按下去时候的坐标+元素初始情况下的 offetLeft.还有一点也是原理性的东西，也就是拖拽的同时是绝对定位，我们改变的是绝对定位条件下的 left 以及 top 等等值。补充：也可以通过 html5 的拖放（Drag 和 drop）来实现

## 30.JS 中的垃圾回收机制

必要性：由于字符串、对象和数组没有固定大小，所有当他们的大小已知时，才能对他们进行动态的存储分配。JavaScript 程序每次创建字符串、数组或对象时，解释器都必 须分配内存来存储那个实体。只要像这样动

态地分配了内存，最终都要释放这些内存以便他们能够被再用，否则，JavaScript 的解释器将会消耗完系统中所有可用的内存，造成系统崩溃。这段话解释了为什么需要系统需要垃圾回收，JS 不像 C/C++，他有自己的一套垃圾回收机制（Garbage Collection）。JavaScript 的解释器可以检测到何时程序不再使用一个对象了，当他确定了一个对象是无用的时候，他就知道不再需要这个对象，可以把它所占用的内存释放掉了。例如：

var a="hello world"; var b="world";

var a=b;

//这时，会释放掉"hello world"，释放内存以便再引用垃圾回收的方法：标记清除、计数引用。

标记清除

这是最常见的垃圾回收方式，当变量进入环境时，就标记这个变量为”进入环境“,从逻辑上讲，永远不能释放进入环境的变量所占的内存，永远不能释放进入环境变量所占用的内存，只要执行流程进入相应的环境，就可能用到他们。当离开环境时，就标记为离开环境。

垃圾回收器在运行的时候会给存储在内存中的变量都加上标记（所有都加），然后去掉环境变量中的变量，以及被环境变量中的变量所引用的变量（条件性去除标记），删除所有被标记的变量，删除的变量无法在环境变量中被访问所以会被删除，最后垃圾回收器，完成了内存的清除工作，并回收他们所占用的内存。

引用计数法

另一种不太常见的方法就是引用计数法，引用计数法的意思就是每个值没引用的次数， 当声明了一个变量，并用一个引用类型的值赋值给改变量，则这个值的引用次数为 1,；相反的，如果包含了对这个值引用的变量又取得了另外一个值，则原先的引用值引用次数就减 1，当这个值的引用次

数为 0 的时候，说明没有办法再访问这个值了，因此就把所占的内存给回收进来，这样垃圾收集器再次运行的时候，就会释放引用次数为 0 的这些值。

用引用计数法会存在内存泄露，下面来看原因：

```
function problem() {
var objA = new Object(); var objB = new Object();
objA.someOtherObject = objB; objB.anotherObject = objA;
}
```

在这个例子里面，objA 和 objB 通过各自的属性相互引用，这样的话，两个对象的引用次数都为 2，在采用引用计数的策略中，由于函数执行之后，这两个对象都离开了作用域，函数执行完成之后，因为计数不为 0，这样的相互引用如果大量存在就会导致内存泄露。

特别是在 DOM 对象中，也容易存在这种问题：

```
var element=document.getElementById（‘‘）; var myObj=new Object();
myObj.element=element; element.someObject=myObj;
```

这样就不会有垃圾回收的过程。

## 31.JS 监听对象属性的改变

参考回答：

我们假设这里有一个 user 对象，

在 ES5 中可以通过 Object.defineProperty 来实现已有属性的监听

```
Object.defineProperty(user,'name',{ set：function(key,value){
}
})
```

缺点：如果 id 不在 user 对象中，则不能监听 id 的变化(2)在 ES6 中可以通过 Proxy 来实现

varuser = new Proxy({}, {

set: function(target,key,value,receiver){

}

})

这样即使有属性在 user 中不存在，通过 user.id 来定义也同样可以这样监听这个属性的变化哦。

## 32.自己实现一个 bind 函数

原理: 通过 apply 或者 call 方法来实现。(1)初始版本

```
Function.prototype.bind=func
tion(obj,arg){
var arg=Array.prototype.slice.call(arguments,1);
var context=this;
return
function(newArg){ arg=arg.concat(Arr
ay.prototype.slice.call(newArg));
return context.apply(obj,arg);
}
}
```

⑵ 考虑到原型链

为什么要考虑？因为在 new 一个 bind 过生成的新函数的时候，必须的条件是要继承原函数的原型

```
Function.prototype.bind=function(obj,arg){
```

```
var
arg=Array.prototype.slice.call(argu
ments,1); var context=this;
var
bound=function(newArg){ arg=arg.co
ncat(Array.prototype.slice.call(newArg)
); return context.apply(obj,arg);
}
var F=function(){}
//这里需要一个寄生组
合继承
F.prototype=context.
prototype;
bound.prototype=ne
w F(); return bound;
}
```

**33.JS 怎么控制一次加载一张图片，加载完后再加载下一张**

(1)方法 1 <script type="text/javascript"> var obj=new Image();
obj.src="http://www.phpernote.com/uploadfiles/editor/20110724050220117 9.jpg";
obj.onload=function()

```
{
alert('图片的宽度为:
'+obj.width+'; 图片的
高度为: '+obj.height);
document.getElemen
tById("mypic").innner
HTML="<img
src='"+this.src+"'
/>";
}
</script>
<div
id="mypic">onloadi
ng.....</div>
```

方法 2

```
<script
type="text/javascript
"> var obj=new
Image();
obj.src="http://www.
phpernote.com/uplo
adfiles/editor/20110
7240502201179.jpg";
obj.onreadystatecha
nge=function(){
if(this.readyState=="
complete")
```

```
alert('图片的宽度为:
'+obj.width+'; 图片的
高度为: '+obj.height);
document.getElemen
tById("mypic").innner
HTML="<img
src='"+this.src+"'
/>";
}
}
</script>
<div
id="mypic">onloadi
ng.....</div>
```

**34.实现 JS 中所有对象的深度克隆（包装对象，Date 对象，正则对象）**

通过递归可以简单实现对象的深度克隆，但是这种方法不管是 ES6 还是 ES5 实现，都有同样的缺陷，就是只能实现特定的 object 的深度复制（比如数组和函数），不能实现包装对象 Number，String ，Boolean，以及 Date 对象，RegExp 对象的复制。

⑴ 前文的方法

```
function deepClone(obj){
var newObj= obj instanceof
Array?[]:{}; for(var i in obj){
newObj[i]=typeof
obj[i]=='object'?
```

```
deepClone(obj[i]):obj[i];
}
return newObj;
}
```

这种方法可以实现一般对象和数组对象的克隆，比如：

```
var arr=[1,2,3];
var newArr=deepClone(arr);
//
newArr->[1,2,
3] var obj={
x:1,
y:2
}
var newObj=deepClone(obj);
// newObj={x:1,y:2}
```

但是不能实现例如包装对象 Number,String,Boolean,以及正则对象 RegExp 和 Date 对象的克隆，比如：

```
//Number 包装对象
var num=new Number(1); typeof num // "object"
var newNum=deepClone(num);
//newNum -> {} 空对象

//String 包装对象
var str=new String("hello"); typeof str //"object"
var newStr=deepClone(str);
//newStr-> {0:'h',1:'e',2:'l',3:'l',4:'o'};

//Boolean 包装对象
var bol=new Boolean(true); typeof bol //"object"
```

```
var newBol=deepClone(bol);
// newBol ->{} 空对象
```

....

⑵ valueof()函数

所有对象都有 valueOf 方法，valueOf 方法对于：如果存在任意原始值，它就默认将对象转换为表示它的原始值。对象是复合值，而且大多数对象无法真正表示为一个原始值， 因此默认的 valueOf()方法简单地返回对象本身，而不是返回一个原始值。数组、函数和正则表达式简单地继承了这个默认方法，调用这些类型的实例的 valueOf()方法只是简单返回这个对象本身。

对于原始值或者包装类： function baseClone(base){ return base.valueOf();
}

```
//Number
var num=new Number(1);
var newNum=baseClone(num);
//newNum->1
//String
var str=new
String('hello'); var
newStr=baseClone
(str);
// newStr->"hello"
//Boolean
var bol=new
Boolean(true); var
```

```
newBol=baseClone(
bol);
//newBol-> true
```

其实对于包装类，完全可以用=号来进行克隆，其实没有深度克隆一说， 这里用 valueOf 实现，语法上比较符合规范。

对于 Date 类型：

因为 valueOf 方法，日期类定义的 valueOf()方法会返回它的一个内部表示：1970 年 1 月 1 日以来的毫秒数.因此我们可以在 Date 的原型上定义克隆的方法： Date.prototype.clone=function(){

```
return new Date(this.valueOf());
}
var date=new
Date('2010'); var
newDate=date.clon
e();
// newDate-> Fri Jan 01 2010 08:00:00 GMT+0800
```

对于正则对象 RegExp：

```
RegExp.prototype.clone =
function() { var pattern =
this.valueOf();
var flags = '';
flags += pattern.global ? 'g' : '';
```

```
flags +=
pattern.ignoreCase ? 'i' : '';
flags += pattern.multiline ?
'm' : '';
return new RegExp(pattern.source, flags);
};
var reg=new
RegExp('/111/'); var
newReg=reg.clone()
;
//newReg->   /\/111\//
```

## 35.来讲讲 JS 的闭包吧

闭包是指有权访问另外一个函数作用域中的变量的函数。
闭包就是函数的局部变量集合，只是这些局部变量在函数返回后会继续存在。闭包就是就是函数的“堆栈”在函数返回后并不释放，我们也可以理解为这些函数堆栈并不在栈上分配而是在堆上分配。当在一个函数内定义另外一个函数就会产生闭包。
（2）为什么要用：
匿名自执行函数：我们知道所有的变量，如果不加上 var 关键字，则默认的会添加到全局对象的属性上去，这样的临时变量加入全局对象有很多坏处，比如：别的函数可能误用这些变量；造成全局对象过于庞大，影响访问速度(因为变量的取值是需要从原型链上遍历的)。除了每次使用变量都是用 var 关键字外，我们在实际情况下经常遇到这样一种情况，即有的函数只需要执行一次，其内部变量无需维护，可以用闭包。
结果缓存：我们开发中会碰到很多情况，设想我们有一个处理过程很

耗时的函数对象， 每次调用都会花费很长时间，那么我们就需要将计算出来的值存储起来，当调用这个函数的时候，首先在缓存中查找，如果找不到，则进行计算，然后更新缓存并返回值，如果找到了，直接返回查找到的值即可。闭包正是可以做到这一点，因为它不会释放外部的引用，从而函数内部的值可以得以保留。

## 36.能来讲讲 JS 的语言特性吗

运行在客户端浏览器上；

不用预编译，直接解析执行代码； 是弱类型语言，较为灵活；

与操作系统无关，跨平台的语言； 脚本语言、解释性语言

## 37.JS 的全排列

```
function permutate(str) {
var result =
[];
if(str.length
> 1) { var
left = str[0];
var rest = str.slice(1,
str.length); var
preResult =
permutate(rest);
for(var i=0;
i<preResult.length; i++)
{ for(var j=0;
j<preResult[i].length; j++) {
```

```
var tmp = preResult[i],slice(0, j) + left + preResult[i].slice(j,
preResult[i].length);result.push(tmp);
}
}
} else if (str.length
== 1) { return
[str];
}
return result;
}
```

# jQuery 相关的知识

## 1.Css 预处理 sass less 是什么？为什么使用他们

Sass 和 less 都是 css 预处理器，是 css 上的一种抽象层，是一种特殊的语法，最终会编译成 css，less 是一种动态样式语言，给 css 赋予了动态

语言的特性，比如：变量，继承，嵌套。Less 既可以在客户端运行，在可以在服务端运行（需要借助 node）

## 2.Js 中.call()与.apply()区别

apply：调用一个对象的一个方法，用另一个对象替换当前对象。

call：调用一个对象的一个方法，用另一个对象替换当前对象。

从定义中可以看出，call 和 apply 都是调用一个对象的一个方法，用另一个对象替换当前对象。而不同之处在于传递的参数，apply 最多只能有两个参数——新 this 对象和一个数组 argArray，如果 arg 不是数组则会报错

相同点:两个方法产生的作用是完全一样的。call, apply 作用就是借用别人的方法来调用,就像调用自己的一样.

不同点:方法传递的参数不同

## 3.为什么会造成跨域/请简述同源策略

出现跨域问题的原因:

在前后端分离的模式下，前后端的域名是不一致的，此时就会发生跨域访问问题。在请求的过程中我们要想回去数据一般都是 post/get 请求，所以..跨域问题出现
跨域问题来源于 JavaScript 的同源策略，即只有 协议+主机名+端口号(如存在)相同，则允许相互访问。也就是说 JavaScript 只能访问和操作自己域下的资源，不能访问和操作其他域下的资源。

同源策略 是由 NetScape 提出的一个著名的安全策略。所谓的同源，指的是协议，域名，端口相同。浏览器处于安全方面的考虑，只允许本域名下的接口交互，不同源的客户端脚本，在没有明确授权的情况下，不能读写对方的资源。

# 4.请输出三种减少页面加载时间的方式

1. 优化图片

2. 图像格式的选择（GIF：提供的颜色较少，可用在一些对颜色要求不高的地方）

3. 优化 CSS（压缩合并 css，如 margin-top, margin-left...）

4. 网址后加斜杠（如 www.campr.com/目录，会判断这个目录是什么文件类型，或者是目录。） cdn 托管

5. 标明高度和宽度（如果浏览器没有找到这两个参数，它需要一边下载

图片一边计算大小，如果图片很多，浏览器需要不断地调整页面。这不但影响速度，也影响浏览体验。

当浏览器知道了高度和宽度参数后，即使图片暂时无法显示，页面上也会腾出图片的空位，然后继续加载后面的内容。从而加载时间快了，浏览体验也更好了）

6. 减少 http 请求（合并文件，合并图片）

# 5.This 指向

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

对象中的 this → 指向其本身

事件中 this → 指向事件对象

# 6.什么是 jsonp 工作原理是什么？他为什么不是真正的 ajax

Jsonp 其实就是一个跨域解决方案。
Js 跨域请求数据是不可以的，但是 js 跨域请求 js 脚本是可以的。
所以可以把要请求的数据封装成一个 js 语句，做一个方法的调用。
跨域请求 js 脚本可以得到此脚本。得到 js 脚本之后会立即执行。
可以把数据做为参数传递到方法中。就可以获得数据。从而解决跨域问题。
jsonp 原理：(动态创建 script 标签，回调函数)
浏览器在 js 请求中，是允许通过 script 标签的 src 跨域请求，可以在请求的结果中添加回调方法名，在请求页面中定义方法，就可获取到跨域请求的数据。

为什么不是真正的 ajax?
1、ajax 和 jsonp 这两种技术在调用方式上"看起来"很像，目的也一样，都是请求一个 url，然后把服务器返回的数据进行处理，因此 jquery 和 ext 等框架都把 jsonp 作为 ajax 的一种形式进行了封装；
2、但 ajax 和 jsonp 其实本质上是不同的东西。ajax 的核心是通过 XmlHttpRequest 获取本页内容，而 jsonp 的核心则是动态添加<script>标签来调用服务器提供的 js 脚本。
3、所以说，其实 ajax 与 jsonp 的区别不在于是否跨域，ajax 通过服务端代理一样可以实现跨域，jsonp 本身也不排斥同域的数据的获取。
4、还有就是，jsonp 是一种方式或者说非强制协议，如同 ajax 一样，它

也不一定非要 json 格式来传递数据，如果你愿意，字符换也行，只不过这样不利于 jsonp 提供公开服务。

## 7.请掌握 2 种以上数组去重的方式

使用 indexof（）方法
使用 lastindexof（）方法　和 indexof 方法一样　indexof 从头部开始匹配　lastindexof 从尾部匹配
ES6 的 set 结构　set 不接受重复数据
使用 sort 方法先将原数组排序，然后与相邻的比较，如果不同则存入新数组
使用 filiter 和 indexof 方法
使用 ES6 的 set 和扩展运算符
使用 set 和 Array.from（）方法　　array.from 可以将 set 结构转成数组
用 splice 和双层循环
使用 includes 方法

## 8.深浅拷贝是什么如何实现？

深拷贝：指针赋值，并且内容拷贝、浅拷贝：只是简单的指针赋值 数组浅拷贝：如果是数组，可以使用数组的一些方法实现：slice()，concat()返回一个新数组的特性实现拷贝。用扩展运算符 spread 实现数组深拷贝：

JSON.parse(JSON.stringify())不仅适用于数组还适用于对象。不能拷贝函数，undefined，symbol。

## 9.为什么 js 是弱类型语言

弱类型语言实现相对于强类型语言来说的，在强类型语言中，变量类型有多种，比如 int char float Boolean 不同类型相互转换有时需要强制转换，而 jacascript 只有一种类型 var，为变量赋值时会自动判断类型并转换，所以是弱类型语言。

## 10.怎么转换 less 为 css

1.用 node 将 less 文件生成为 css 文件
2.用 webstorm 自动生成

## 11.echarts 使用最多的是什么

图表及图表组合

# 12.For 循环与 map 循环有什么区别

For 遍历对象自身的和继承可枚举的属性，也就是说会包括哪些原型链上的属性
Map 方法不会对空数组进行检测，map 会返回一个新数组，不会对原数组产生影响

# 13.请写出一个简单的类与继承

创建类有三种：
使用 function 和 this 关键字
原型方法　用 prototype 和 this 关键字
使用 object.create()方法构造
继承有六种：
原型继承　　借用构造函数继承　　组合继承　　原型式继承　　寄生式继承　　寄生组合式继承

# 14.同步与异步的区别/阻塞与非阻塞区别

同步（阻塞的）
异步（非阻塞）
比如：同步，咱两在一起上班，到吃饭时间了，我去喊你一起吃饭，你很忙，我就坐着等你忙完再一起去吃饭

异步，咱两在一起上班，到吃饭时间了，我去喊你一起吃饭，你很忙，我就先自己去吃了，你忙完了再去吃饭
同步（阻塞）异步（非阻塞）这两个关注的是程序在等待调用结果时的状态

# 15.重绘和回流是什么

回流：当 render tree 中的一部分或者全部因为元素的规模尺寸，布局，隐藏等改变而需要重新构建，这就叫回流，每个页面至少需要一次回流，就是在页面第一次加载的时候，这时候一定会发生回流，因为要构建 render tree
在回流的时候，浏览器会使渲染树中收到影响的部分失效，并重新构造这部分渲染树，完成回流后，浏览器会重新绘制受影响的部分到屏幕中，这就是重绘
当 render tree 中的一些元素需要更新属性，而这些属性只是影响元素的外观，不会影响布局，就叫重绘

# 16.http 是什么？有什么特点

http 叫做超文本传输协议，是互联网应用最广泛的一种网络协议
特点：基于请求-响应的模式　无状态保存　无连接

# 17.HTTP 协议和 HTTPS 区别

http 是超文本传输协议，信息是明文传输，https 是具有安全性的 ssl 解密传输协议
http 和 https 连接方式完全不同，端口也不同，http 是 80，https 是 443
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

## 18.原型和继承，prototype，call 和 apply 继承的区别（第一个参数是相同的，第二个的区别在哪）

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。

## 19.箭头函数与普通函数的区别

箭头函数是匿名函数，不能作为构造函数，不能使用 new
箭头函数不能绑定 arguments，要用 rest 参数解决
箭头函数没有原型属性
箭头函数的 this 永远指向其上下文的 this，
箭头函数不能绑定 this，会捕获其所在的上下文的 this 值，作为自己的 this 值

# 20.什么是 js 内存泄露？

内存泄漏是指一块被分配的内存既不能使用又不能回收，直到浏览器进程结束
释放内存的方法：赋值为 null

# 21.你如何对网站的文件和资源进行优化？

1、文件合并（目的是减少 http 请求）
2、文件压缩（目的是直接减少文件下载的体积）
3、使用 cdn 托管资源
4、使用缓存
5、gizp 压缩你的 js 和 css 文件
6、meta 标签优化（title,description,keywords）、heading 标签的优化、alt 优化　　7、反向链接，网站外链接优化

# 22.请简述 ajax 的执行过程 以及常见的 HTTP 状态码

100：这个状态码是告诉客户端应该继续发送请求，这个临时响应是用来通知客户端的，部分的请求服务器已经接受，但是客户端应继续发送求请求的剩余部分，如果请求已经完成，就忽略这个响应，而且服务器会在请求完成后向客户发送一个最终的结果

200：这个是最常见的 http 状态码，表示服务器已经成功接受请求，并将返回客户端所请求的最终结果

202：表示服务器已经接受了请求，但是还没有处理，而且这个请求最终会不会处理还不确定

204：服务器成功处理了请求，但没有返回任何实体内容 ，可能会返回新的头部元信息

301：客户端请求的网页已经永久移动到新的位置，当链接发生变化时，返回 301 代码告诉客户端链接的变化，客户端保存新的链接，并向新的链接发出请求，已返回请求结果

404：请求失败，客户端请求的资源没有找到或者是不存在

500：服务器遇到未知的错误，导致无法完成客户端当前的请求。

503：服务器由于临时的服务器过载或者是维护，无法解决当前的请求

# 23.预加载和懒加载的区别，预加载在什么时间加载合适

预加载是指在页面加载完成之前，提前将所需资源下载，之后使用的时候从缓存中调用；懒加载是延迟加载，按照一定的条件或者需求等到满足条件的时候再加载对应的资源

两者主要区别是一个是提前加载，一个是迟缓甚至不加载。懒加载对服务器前端有一定的缓解压力作用，预加载则会增加服务器前端压力。

# 24.Jquery 选择器有哪些

一、基本选择器

基本选择器是 jQuery 中最常用也是最简单的选择器，它通过元素的 id、class 和标签名等来查找 DOM 元素。

1、ID 选择器 #id

描述：根据给定的 id 匹配一个元素， 返回单个元素（注：在网页中，id 名称不能重复）

示例：$("#test") 选取 id 为 test 的元素

2、类选择器 .class

描述：根据给定的类名匹配元素，返回元素集合

示例：$(".test") 选取所有 class 为 test 的元素

3、元素选择器 element

描述：根据给定的元素名匹配元素，返回元素集合

示例：$("p") 选取所有的<p>元素

4、*

描述：匹配所有元素，返回元素集合

示例：$("*") 选取所有的元素

5、selector1，selector2,...,selectorN

描述：将每个选择器匹配到的元素合并后一起返回，返回合并后的元素集合

示例：$("p,span,p.myClass") 选取所有<p>,和 class 为 myClass 的<p>标签的元素集合

二、层次选择器

层次选择器根据层次关系获取特定元素。

1、后代选择器

示例：$("p span") 选取<p>元素里的所有的元素（注：后代选择器选择父元素所有指定选择的元素，不管是儿子级，还是孙子级）

2、子选择器 $("parent>child")

示例：$("p>span") 选择<p>元素下的所有元素 （注：子选择器只选择直属于父元素的子元素）

3、同辈选择器 $("prev+next")

描述：选取紧接在 prev 元素后的 next 元素，返回元素集合

示例：$(".one+p") 选取 class 为 one 的下一个<p>同辈元素集合

4、同辈选择器 $("prev~siblings")

描述：选取 prev 元素后的所有 siblings 元素，返回元素集合

示例：$("#two~p")选取 id 为 two 的元素后所有<p>同辈元素集合

三、过滤选择器

1>基本过滤选择器

1、 :first

描述：选取第一个元素，返回单个元素

示例：$("p:first") 选取所有<p>元素中第一个<p>元素

2、 :last

描述：选取最后一个元素，返回单个元素

示例：$("p:last") 选取所有<p>元素中最后一个<p>元素

3、 :not(selector)

描述：去除所有与给定选择器匹配的元素，返回元素集合

示例：$("input:not(.myClass)") 选取 class 不是 myClass 的<input>元素

4、 :even

描述：选取索引是偶数的所有元素，索引从 0 开始，返回元素集合

5、 :odd

描述：选取索引是奇数的所有元素，索引从 0 开始，返回元素集合

6、 :eq(index)

描述：选取索引等于 index 的元素，索引从 0 开始，返回单个元素

7、 :gt(index)

描述：选取索引大于 index 的元素，索引从 0 开始，返回元素集合

8、 :lt(index)

描述：选取索引小于于 index 的元素，索引从 0 开始，返回元素集合

9、 :focus

描述：选取当前获取焦点的元素

2>内容过滤选择器

1、:contains(text)

描述：选取含有文本内容为 text 的元素，返回元素集合

示例：$("p:contains('我')") 选取含有文本“我”的元素

2、:empty

描述：选取不包含子元素或者文本元素的空元素，返回元素集合

示例：$("p:empty")选取不包含子元素或者文本元素的空 元素（ ）

3、:has(selector)

描述：选取含有选择器所匹配的元素的元素，返回元素集合

示例：$("p:has(p)")选取含有 元素的 元素（ ）

4、:parent

描述：选取含有子元素或者文本的元素，返回元素集合

示例：$("p:parent")选取含有子元素或者文本元素的 元素（ 或者 文本 ）

3>可见性过滤选择器

1、:hidden

描述：选取所有不可见的元素，返回元素集合

2、:visible

描述：选取所有可见的元素，返回元素集合

4>属性过滤选择器（返回元素集合）

1、[attribute]

示例：$("p[id]")选取拥有 id 属性的 p 元素

2、[attribute=value]

示例：$("input[name=text]")选取拥有 name 属性等于 text 的 input 元素

3、[attribute!=value]

示例：$("input[name!=text]")选取拥有name属性不等于text的input元素

4、[attribute^=value]

示例：$("input[name^=text]")选取拥有 name 属性以 text 开始的 input 元素

5、[attribute$=value]

示例：$("input[name$=text]")选取拥有 name 属性以 text 结束的

input 元素
6、[attribute*=value]
示例：$("input[name*=text]")选取拥有 name 属性含有 text 的 input 元素
7、[attribute~=value]
示例：$("input[class~=text]")选取拥有 class 属性以空格分割的值中含有 text 的 input 元素
8、[attribute1][attribute2][attributeN]
描述：合并多个属性过滤选择器
5>表单对象属性过滤选择器（返回元素集合）
1、:enabled
描述：选取所有可用元素
2、:disabled
描述：选取所有不可用元素
3、:checked
描述：选取所有被选中的元素（单选框，复选框）
示例：$("input:checked")选取所有被选中的<input>元素
4、:selected
描述：选取所有被选中的选项元素（下拉列表）
示例：$("select option:selected")选取所有被选中的选项元素

# 25.Jquery 插入节点的方法

append() 向每个匹配的元素内部追加内容
appendTo() 将所有匹配的元素追加到指定元素中，实际上，使用该方

法是颠倒了常规的$(A).append(B)的操作 将 A 追加到 B 中
prepend() 向每个匹配的元素内部前置内容
prependTo() 将所有匹配的元素前置到指定的元素中。实际上，使用该方法是颠倒了常规的$(A).prepend(B)的操作，即不是将 B 前置到 A 中，而是将 A 前置到 B 中
after() 在每个匹配的元素之后插入内容
insertAfter() 将所有匹配的元素插入到指定元素的后面。实际上，使用该方法是颠倒了常规的$(A).after(B)的操作，即不是讲 B 插入到 A 后面，而是将 A 插入到 B 后面
before() 在每个匹配的元素之前插入内容
insertBefore() 将所有匹配的元素插入到指定的元素的前面。实际上，使用该方法是颠倒了常规的$(A).before(B)的操作，即不是将 B 插入到 A 前面，而是将 A 插入到 B 前面

## 26.Js 的函数节流和函数防抖的区别

函数节流是指一定时间内 js 方法只执行一次。
函数防抖是指频繁触发的情况下，只有足够的空闲时间，才执行代码一次
函数节流是 声明一个变量当标志位，记录当前代码是否在执行，如果正在执行，取消这次方法执行，直接 return，如果空闲，正常触发方法执行
函数防抖是需要一个延时器来辅助实现，延迟执行需要执行的代码，如果方法多次触发，把上次记录的延迟执行代码用 cleartimeout 清除掉，重新开始，如果计时完毕，没有方法来访问触发，则执行代码、

## 27.Get 和 post 不同

Get 是从服务器上获取数据，post 是向服务器传送数据
在客户端，get 通过 url 提交数据，数据在 url 中可以看到，post 方式，数据放在 html header 中提交
安全性问题
Get 提交数据最多只能有 1024 字节，post 没有限制

## 28.什么是 csrf 攻击

Csrf（跨站点请求伪造） 攻击者在用户已经登录目标网站之后，诱使用户访问一个攻击页面，利用目标网站对用户的信任，以用户身份再攻击页面对目标网站发起伪造用户操作的请求，达到攻击目的
Js 中手写一个深拷贝

## 29.什么时候用深拷贝 /浅拷贝

无论深浅，都是需要的，当深拷贝发生时通常表明存在着一个聚合关系，

当浅拷贝发生时，通常表明存在着相识关系
举个简单例子：当实现一个组合模式 Composite Pattern 时通常都会实现深拷贝
当实现一个观察者模式 Observer Pattern，时，就需要实现浅拷贝

# Vue 相关

## 1.Vue 的核心是什么

Vue 是一套构建用户界面的渐进式自底向上增量开发的 MVVM 框架，vue 的核心只关注视图层，
核心思想：
数据驱动（视图的内容随着数据的改变而改变）
组件化（可以增加代码的复用性，可维护性，可测试性，提高开发效率，方便重复使用，体现了高内聚低耦合）

## 2.请简述你对 vue 的理解

Vue 是一套构建用户界面的渐进式的自底向上增量开发的 MVVM 框架，核心是关注视图层，vue 的核心是为了解决数据的绑定问题，为了开发大型单页面应用和组件化，所以 vue 的核心思想是数据驱动和组件化，这

里也说一下 MVVM 思想，MVVM 思想是 模型 视图 vm 是 v 和 m 连接的桥梁，当模型层数据修改时，VM 层会检测到，并通知视图层进行相应修改

## 3.请简述 vue 的单向数据流

父级 prop 的更新会向下流动到子组件中，每次父组件发生更新，子组件所有的 prop 都会刷新为最新的值
数据从父组件传递给子组件，只能单向绑定，子组件内部不能直接修改父组件传递过来的数据，（可以使用 data 和 computed 解决）

## 4. Vue 常用的修饰符有哪些

修饰符：.lazy 改变后触发，光标离开 input 输入框的时候值才会改变
.number 将输出字符串转为 number 类型
.trim 自动过滤用户输入的首尾空格
事件修饰符：
.stop 阻止点击事件冒泡，相当于原生 js 中的 event.stopPropagation()
.prevent 防止执行预设的行为，相当于原生 js 中 event.preventDefault()
.capture 添加事件侦听器时使用事件捕获模式，就是谁有该事件修饰符，就先触发谁

.self 只会触发自己范围内的事件，不包括子元素
.once 只执行一次
键盘修饰符：
.enter 回车键 .tab 制表键 .esc 返回键 .space 空格键
.up 向上键 .down 向下键 .left 向左建 .right 向右键
系统修饰符：.ctrl .alt .shift .meta

## 5.v-text 与{{}}与 v-html 区别

{{}} 将数据解析为纯文本，不能显示输出 html
v-html 可以渲染输出 html
v-text 将数据解析为纯文本，不能输出真正的 html，与花括号的区别是在页面加载时不显示双花括号
v-text 指令：
作用：操作网页元素中的纯文本内容。{{}}是他的另外一种写法
v-text 与{{}}区别：
v-text 与{{}}等价，{{}}叫模板插值，v-text 叫指令。
有一点区别就是，在渲染的数据比较多的时候，可能会把大括号显示出来，俗称屏幕闪动：

## 6.v-on 可以绑定多个方法吗

可以　如果绑定多个事件，可以用键值对的形式 事件类型：事件名
　　如果绑定是多个相同事件，直接用逗号分隔就行

## 7.Vue 循环的 key 作用

Key 值的存在保证了唯一性，Vue 在执行时，会对节点进行检查，如果没有 key 值，那么 vue 检查到这里有 dom 节点，就会对内容清空并赋新值，如果有 key 值存在，那么会对新老节点进行对比，比较两者 key 是否相同，进行调换位置或删除操作

## 8.什么是计算属性

计算属性是用来声明式的描述一个值依赖了其他的值，当它依赖的这个值发生改变时，就更新 DOM
当在模板中把数据绑定到一个计算属性上时，vue 会在它依赖的任何值导致该计算属性改变时更新 DOM
每个计算属性都包括一个 getter 和 setter，读取时触发 getter，修改时触发 setter

# 9. Vue 单页面的优缺点

单页面 spa
优点：前后端分离 用户体验好 一个字 快 内容改变不需要重新加载整个页面
缺点：不利于 seo， 初次加载时耗长（浏览器一开始就要加载 html css js ，所有的页面内容都包含在主页面中） ，页面复杂度提高了，导航不可用

# 10.Vuex 是什么？怎么使用？在那种场景下使用

Vuex 是一个专为 vue.js 应用程序开发的状态管理模式，通过创建一个集中的数据存储，方便程序中的所有组件进行访问，简单来说 vuex 就是 vue 的状态管理工具

Vuex 有五个属性 state getters mutations actions modules
State 就是数据源存放地，对应一般 vue 对象的 data，state 里面存放的数据是响应式的，state 数据发生改变，对应这个数据的组件也会发生改变 用 this.$store.state.xxx 调用
Getters 相当于 store 的计算属性，主要是对 state 中数据的过滤，用 this.$store.getters.xxx 调用
Mutations 处理数据逻辑的方法全部放在 mutations 中，当触发事件想

改变 state 数据的时候使用 mutations，用 this.$store.commit 调用，给这个方法添加一个参数，就是 mutation 的载荷（payload）
Actions 异步操作数据，但是是通过 mutation 来操作 用 this.$store.dispatch 来触发，actions 也支持载荷
使用场景：组件之间的状态，登录状态，加入购物车，音乐播放
Vuex 使用流程：

下载 vuex

在 src 下创建 store 以及 index.js
引入 vue 和 vuex， 使用 vuex ，导出实例对象
在 main.js 中引入，在.vue 文件中使用

## 11.Vue 中路由跳转方式（声明式/编程式）

Vue 中路由跳转有两种，分别是声明式和编程式

用 js 方式进行跳转的叫编程式导航 this.$router.push（）

用 router-link 进行跳转的叫声明式 router-view 路由出口，路由模板显示的位置

路由中 name 属性有什么作用?
在 router-link 中使用 name 导航到对应路由
使用 name 导航的同时，给子路由传递参数

# 12.vue 跨域的解决方式

1.后台更改 header
2.使用 jq 提供 jsonp
3.用 http-proxy-middleware（配置代理服务器的中间件）

# 13.Vue 的生命周期请简述

vue 的生命周期就是 vue 实例创建到实例销毁的过程。期间会有 8 个钩子函数的调用。
beforeCreate（创建实例）
created（创建完成）、
beforeMount（开始创建模板）
mounted（创建完成）、
beforeUpdate（开始更新）
updated（更新完成）、
beforeDestroy（开始销毁）
destroyed（销毁完成）

# 14.Vue 生命周期的作用

给了用户在不同阶段添加自己的代码的机会

## 15.DOM 渲染在那个生命周期阶段内完成

DOM 渲染在 mounted 周期中就已经完成

## 16.Vue 路由的实现

前端路由就是更新视图但不请求页面，

利用锚点完成切换，页面不会刷新
官网推荐用 vue-router.js 来引入路由模块
定义路由组件
定义路由，使用 component 进行路由映射组件，用 name 导航到对应路由
创建 router 实例，传入 routes 配置
创建和挂载根实例
用 router-link 设置路由跳转

## 17.Vue 路由模式 hash 和 history，简单讲一下

Hash 模式地址栏中有#，history 没有，history 模式下刷新，会出现 404 情况，需要后台配置

使用 JavaScript 来对 loaction.hash 进行赋值，改变 URL 的 hash 值

可以使用 hashchange 事件来监听 hash 值的变化

HTML5 提供了 History API 来实现 URL 的变化。其中最主要的 API 有以下两个：history.pushState() 和 history.repalceState()。这两个 API 可以在不进行刷新的情况下，操作浏览器的历史纪录。唯一不同的是，前者是新增一个历史记录，后者是直接替换当前的历史记录。

## 18.Vue 路由传参的两种方式，params 和 query 方式与区别

动态路由也可以叫路由传参，就是根据不同的选择在同一个组件渲染不同的内容

用法上：query 用 path 引入，params 用 name 引入，接收参数都是类似的，分别是 this.$route.query.name 和 this.$route.params.name

url 展示上:params 类似于 post，query 类似于 get，也就是安全问题，

params 传值相对更安全点，query 通过 url 传参，刷新页面还在，params 刷新页面不在了

## 19.Vue 数据绑定的几种方式

1.单向绑定 双大括号 {{}} html 内字符串绑定
2.v-bind 绑定 html 属性绑定
3.双向绑定 v-model
4.一次性绑定 v-once 依赖于 v-model

## 20.Vue 注册一个全局组件

Vue.componnet（“组件的名字”{对象 template 组建的内容 })

## 21.Vue 的路由钩子函数/路由守卫有哪些

全局守卫：beforeEach（to，from，next）和 afterEach（to，from）

路由独享守卫：beforeEnter
组件内的守卫：路由进入/更新/离开之前
beforeRouterEnter/update/leave

## 22.Vue 中如何进行动态路由设置？有哪些方式？怎么获取传递过来的数据？

动态路由也可以叫路由传参，
动态路由有 query 和 prrams 两种方式传参
query 用 path 引入，params 用 name 引入，query 用 this.$route.query.name 接收参数
params 用 this.$route.params.name 接收参数

## 23.Elementui 中的常用组件有哪些？请简述你经常使用的 并且他们的属性有哪些？

Container 布局容器
<el-container>外层容器

<el-header>顶栏容器
<el-aside>侧边栏容器
<el-main>主要内容容器
<el-footer>底栏容器
Dropdown 下拉菜单
<el-container split-buton> 下拉按钮
<el-container-menu> 下拉菜单
<el-container-item> 下拉项
Table 表格
Tabs 标签页
Form 表单
Pagination 分页
Message 消息提示

## 24.Vue-cli 中如何自定义指令

## 25.Vue 中指令有哪些

v-for：循环数组，对象，字符串，数字
v-on：绑定事件监听

v-bind：动态绑定一个或者多个属性
v-model：表单控件或者组件上创建双向绑定
v-if　v-else　v-else-if　条件渲染
v-show　根据表达式真假，切换元素的 display
v-html 更新元素的 innerhtml
v-text 更新元素的 textcontent
v-pre 跳过这个元素和子元素的编译过程
v-clock 这个指令保持在元素上知道关联实例结束编译
v-once　只渲染一次

## 26.Vue 如何定义一个过滤器

过滤器本质就是一个有参数有返回值的方法

```
new Vue({
        filters:{
            myCurrency:function(myInput){
                return 处理后的数据
            }
        }
})
```

使用方法：<h1>{{表达式 | 过滤器}}</h1>
过滤器高级用法：可以指定参数，告诉过滤器按照参数进行数据的过滤

## 27.对 vue 中 keep-alive 的理解

概念：keep-alive 是 vue 的内置组件，当它动态包裹组件时，会缓存不活动的组件实例，它自身不会渲染成一个 DOM 元素也不会出现在父组件链中
作用：在组件切换过程中将状态保留在内存中，防止重复渲染 DOM，减少加载时间以及性能消耗，提高用户体验。
生命周期函数：Activated 在 keep-alive 组件激活时调用，deactivated 在 keep-alive 组件停用时调用

## 28.如何让组件中的 css 在当前组件生效

在 styled 中加上 scoped

## 29.Vue 生命周期一共几个阶段

创建 加载 更新 销毁
Beforecreate 创建前
Created 创建后

Beforemount 加载前
Mounted 加载后
Beforeupdate 更新前
Updated 更新后
Beforedestroy 销毁前
Destroyed 销毁后
页面第一次加载会触发 beforecreate created beforemount mounted
DOM 渲染在 mounted 周期中就已经完成

## 30.Mvvm 与 mvc 的区别

Mvc 模型视图控制器，视图是可以直接访问模型，所以，视图里面会包含模型信息，mvc 关注的是模型不变，所以，在 mvc 中，模型不依赖视图，但是视图依赖模型
Mvvm 模型 视图 和 vm vm 是作为模型和视图的桥梁，当模型层数据改变，vm 会检测到并通知视图层进行相应的修改

## 31.Vue 组件中的 data 为什么是函数

Data 是一个函数时，每个组件实例都有自己的作用域，每个实例相互独立，不会相互影响

如果是引用类型（对象），当多个组件共用一个数据源时，一处数据改变，所有的组件数据都会改变，所以要利用函数通过 return 返回对象的拷贝，（返回一个新数据），让每个实例都有自己的作用域，相互不影响。

## 32.Vue 双向绑定的原理

Vue 双向绑定就是：数据变化更新视图，视图变化更新数据
Vue 数据双向绑定是通过数据劫持和观察者模式来实现的，
数据劫持，object.defineproperty 它的目的是：当给属性赋值的时候，程序可以感知到，就可以控制改变属性值
观察者模式 当属性发生改变的时候，使用该数据的地方也发生改变

## 33.Vue 中组件怎么传值

正向：父传子 父组件把要传递的数据绑定在属性上，发送，子组件通过 props 接收
逆向：子传父 子组件通过 this.$emit（自定义事件名，要发送的数据），父组件设置一个监听事件来接收，然后拿到数据
兄弟：eventbus 中央事件总线
通过 Vuex

# 34.Bootstrap 的原理

网格系统的实现原理，通过定义容器大小，平分 12 份，（24 份或者 32 份），再调整内外边距，结合媒体查询，就成了强大的响应式网格系统。

比如 row col-xs-4

36.如果一个组件在多个项目中使用怎么办

## 37.槽口请简述

大概分这几点，首先槽口（插槽）可以放什么内容？放在哪？什么作用？可以放任意内容，在子组件中使用，是为了将父组件中的子组件模板数据正常显示。

具名插槽和匿名插槽，作用域插槽，说白了就是在组件上的属性，可以在组件元素内使用，

可以在父组件中使用 slot-scope 从子组件获取数据

**38.Watch 请简述**

Watch 的作用是监控一个值的变化，并调用因为变化需要执行的方法

**39.Vant Ui 请简述下**

轻量、可靠的移动端 Vue 组件库

**40.计算属性与 watch 区别**

Computed watch 区别就是 computed 的缓存功能，当无关数据数据改变时，不会重新计算，直接使用缓存中的值。计算属性是用来声明式的描述一个值依赖了其他的值，当所依赖的值后者变量发生变化时，计算属性也跟着改变，
Watch 监听的是在 data 中定义的变量，当该变量变化时，会触发 watch 中的方法

**41.mvvm 框架是什么？它和其它框架（jquery）的区别是什么？哪些场景适合？**

Mvvm 和其他框架的区别是 vue 数据驱动 通过数据来显示视图而不是节点操作

适用于数据操作比较多的场景

## 42.Vue 首屏加载慢的原因，怎么解决的，白屏时间怎么检测，怎么解决白屏问题

首屏加载慢的原因：
第一次加载页面有很多组件数据需要渲染
解决方法：
1.路由懒加载 component：（）=>import(“路由地址”)
2.ui 框架按需加载
3.gzip 压缩
白屏时间检测：
？？？？
解决白屏问题：
①使用 v-text 渲染数据
②使用{{}}语法渲染数据，但是同时使用 v-cloak 指令（用来保持在元素上直到关联实例结束时候进行编译），v-cloak 要放在什么位置呢，v-cloak 并不需要添加到每个标签，只要在 el 挂载的标签上添加就可以

## 43.Vue 双数据绑定过程中，这边儿数据改变了怎么通知另一边改变

数据劫持和观察者模式
Vue 数据双向绑定是通过数据劫持和观察者模式来实现的，
数据劫持，object.defineproperty 它的目的是：当给属性赋值的时候，

程序可以感知到，就可以控制属性值的有效范围，可以改变其他属性的值
观察者模式它的目的是当属性发生改变的时候，使用该数据的地方也发生改变

## 44.Vuex 流程

在 vue 组件里面，通过 dispatch 来触发 actions 提交修改数据的操作，然后通过 actions 的 commit 触发 mutations 来修改数据，mutations 接收到 commit 的请求，就会自动通过 mutate 来修改 state，最后由 store 触发每一个调用它的组件的更新

## 45.Vuex 怎么请求异步数据

1.首先在 state 中创建变量
2.然后在 action 中调用封装好的 axios 请求，异步接收数据，commit 提交给 mutations
Mutations 中改变 state 中的状态，将从 action 中获取到的值赋值给 state

## 46.Vuex 中 action 如何提交给 mutation 的

Action 函数接收一个与 store 实例具有相同方法和属性的 context 对象，

可以调用 context.commit 提交一个 mutation，或者通过 context.state 和 context.getters 获取 state 和 getters

**47.Route 与 router 区别**

1. router 是 VueRouter 的一个对象，通过 Vue.use(VueRouter)和 VueRouter 构造函数得到一个 router 的实例对象，这个对象中是一个全局的对象，他包含了所有的路由包含了许多关键的对象和属性。

2.route 是一个跳转的路由对象，每一个路由都会有一个 route 对象，是一个局部的对象，可以获取对应的 name,path,params,query 等

**49.vuex 的 State 特性是？**

State 就是数据源的存放地
State 里面的数据是响应式的，state 中的数据改变，对应这个数据的组件也会发生改变
State 通过 mapstate 把全局的 state 和 getters 映射到当前组件的计算属性中

**50.vuex 的 Getter 特性是？**

Getter 可以对 state 进行计算操作，它就是 store 的计算属性

Getter 可以在多组件之间复用
如果一个状态只在一个组件内使用，可以不用 getters

**51.vuex 的 Mutation 特性是？**

更改 vuex　store 中修改状态的唯一办法就是提交 mutation，可以在回调函数中修改 store 中的状态

**52.vuex 的 actions 特性是？**

Action 类似于 mutation，不同的是 action 提交的是 mutation，不是直接变更状态，可以包含任意异步操作

**54.vuex 的优势**

优点：解决了非父子组件的通信，减少了 ajax 请求次数，有些可以直接从 state 中获取
缺点：刷新浏览器，vuex 中的 state 会重新变为初始状态，解决办法是 vuex-along，得配合计算属性和 sessionstorage 来实现

## 55.Vue 路由懒加载（按需加载路由）

## 56.v-for 与 v-if 优先级

首先不要把 v-if 与 v-for 用在同一个元素上，原因：v-for 比 v-if 优先，如果每一次都需要遍历整个数组，将会影响速度，尤其是当之需要渲染很小一部分的时候。

v-for 比 v-if 具有更高的优先级

## 57.请写出饿了么 5 个组件

<el-alert>弹窗</el-alert>
<el-dialog>对话</el-dialog>
<el-calender>日历表</el-calender>
<el-progress:percentage="0">进度条<el-progrees>
<el-switch>开关</el-switch>

## 58.vue 在 created 和 mounted 这两个生命周期中请求数据有什么区别呢？

看实际情况，一般在 created（或 beforeRouter） 里面就可以，如果涉及到需要页面加载完成之后的话就用 mounted。
在 created 的时候，视图中的 html 并没有渲染出来，所以此时如果直接去操作 html 的 dom 节点，一定找不到相关的元素

而在 mounted 中，由于此时 html 已经渲染出来了，所以可以直接操作 dom 节点，（此时 document.getelementById 即可生效了）。

## 59.说说你对 proxy 的理解

参考回答：

vue 的数据劫持有两个缺点:

1、无法监听通过索引修改数组的值的变化

2、无法监听 object 也就是对象的值的变化所以 vue2.x 中才会有 $set 属性的存在

proxy 是 es6 中推出的新 api，可以弥补以上两个缺点，所以 vue3.x 版本用 proxy 替换 object.defineproperty。

## 60.Vue3.0 是如何变得更快的？（底层，源码）

a. diff 方法优化
Vue2.x 中的虚拟 dom 是进行全量的对比。
b.Vue3.0 中新增了静态标记（PatchFlag）：在与上次虚拟结点进行对比的时候，值对比带有 patch flag 的节点，并且可以通过 flag 的信息得知当前节点要对比的具体内容化。hoistStatic 静态提升
Vue2.x：无论元素是否参与更新，每次都会重新创建。
Vue3.0：对不参与更新的元素，只会被创建一次，之后会在每次渲染时候被不停的复用。
c.cacheHandlers 事件侦听器缓存
默认情况下 onClick 会被视为动态绑定，所以每次都会去追踪它的变化但是因为是同一个函数，所以没有追踪变化，直接缓存起来复用即可。

# React 相关

## 1.fetch VS ajax VS axios

传统 Ajax 指的是 XMLHttpRequest（XHR），最早出现的发送后端请求技术，隶属于原始 js 中，核心使用 XMLHttpRequest 对象，多个请求之间如果有先后关系的话，就会出现回调地狱。JQuery ajax 是对原生 XHR 的封装
axios 是一个基于 Promise ，本质上也是对原生 XHR 的封装，只不过它是 Promise 的实现版本，符合最新的 ES 规范，

fetch 不是 ajax 的进一步封装，而是原生 js，没有使用 XMLHttpRequest 对象。

## 2.React 事件处理---修改 this 指向

方式 1：通过 bind 方法进行原地绑定，从而改变 this 指向
方式 2：通过创建箭头函数
方式 3：在 constructor 中提前对事件进行绑定
方式 4：将事件调用的写法改为箭头函数的形式

## 3.请简述你对 react 的理解

React 起源于 facebook，react 是一个用于构建用户界面的 js 库

**特点**:

声明式设计：react 采用范式声明，开发者只需要声明显示内容，react 就会自动完成

**高效**： react 通过对 dom 的模拟（也就是虚拟 dom），最大限度的减少与 dom 的交互

**灵活**： react 可以和已知的库或者框架很好配合

**组件**： 通过 react 构建组件，让代码更容易复用，能够很好应用在大型项目开发中，把页面功能拆分成小模块 每个小模块就是组件

**单向数据流**： react 是单向数据流，数据通过 props 从父节点传递到子节点，如果父级的某个 props 改变了，react 会重新渲染所有的子节点

## 4.react 组件之间的数据传递

正向传值用 props

逆向传值用函数传值 通过事件调用函数传递

同级传值用 pubsub-js

用 pubsub.publish（事件名，数据）抛出数据

用 pubsub.subscribe（监听的事件，（）=）{}）接收数据

跨组件传递 用 context 要使用 context 进行跨组件传值就需要使用 createContext() 方法，这个方法有两个对象 provider 生产

者　Consumer 消费者

## 5.Vue 与 react 区别

**相同点**:

都支持服务器渲染

都有虚拟 dom，组件化开发，通过 props 参数进行父子组件数据的传递，

都实现 webcomponent 规范

都是数据驱动视图

都有状态管理，react 有 redux，vue 有 vuex

都有支持 native’的方案 react 有 react native vue 有 weex

**不同点**:

react 严格上只针对 mvc 的 view 层，vue 是 mvvm 模式

虚拟 dom 不一样，vue 会跟踪每一个组件的依赖关系，不需要重新渲染整个 dom 组件树，而 react 不同，当应用的状态被改变时，全部组件都会重新渲染，所以 react 中用 shouldcomponentupdate 这个生命周期的钩子函数来控制

组件写法不一样 ，react 是 jsx 和 inline style ，就是把 html 和 css 全写进 js 中，vue 则是 html，css ，js 在同一个文件

数据绑定不一样，vue 实现了数据双向绑定，react 数据流动是单向的

在 react 中，state 对象需要用 setstate 方法更新状态，在 vue 中，state 对象不是必须的，数据由 data 属性在 vue 对象中管理

## 6.请简述虚拟 dom 与 diff 算法

虚拟 DOM 也就是常说的虚拟节点，它是通过 js 的 object 对象模拟 DOM 中的节点，然后再通过特定的渲染方法将其渲染成真实的 DOM 节点。

频繁的操作 DOM，或大量造成页面的重绘和回流

Diff 算法：把树形结构按照层级分解，只比较同级元素，给列表结构的每个单元添加唯一的 key 值，方便比较

**7.你对组件的理解**

可组合，可复用，可维护，可测试

**8.调用 setState 之后发生了什么？**

React 在调用 setstate 后，react 会将传入的参数对象和组件当前的状态合并，触发调和过程，
在调和过程中，react 会根据新的状态构建 react 元素树重新渲染整个 UI 界面，在得到元素树之后，react 会自动计算新老节点的差异，根据差异对界面进行最小化重新渲染

**9.react 生命周期函数**

componentWillMount 组件渲染之前调用
componentDidMount 在第一次渲染之后调用
componentWillReceiveProps 在组件接收到一个新的 props 时调用
shouldComponentUpdate 判断组件是否更新 html
componentWillupdate 组件即将更新 html 时调用
componentDidupdate 在组件完成更新后立即调用
componentWillUnmount 在组件移除之前调用

## 10.为什么虚拟 dom 会提高性能?(必考)

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

## 11.(组件的)状态(state)和属性(props)之间有何不同

State 与 props 区别

Props 是一个从外部传进组件的参数，主要作用就是父组件向子组件传递数据，但是 props 对于使用它的组件来说是只读的，一旦赋值不能修改，只能通过外部组件主动传入新的 props 来重新渲染子组件
State 一个组件的显示形态可以由数据状态和外部参数决定，外部参数是 props，数据状态就是 state，首先，在组件初始化的时候，用 this.state 给组件设定一个初始的 state，在第一次渲染的时候就会用这个数据来渲染组件，state 不同于 props 一点时，state 可以修改，通过 this.setState() 方法来修改 state

## 12.shouldComponentUpdate 是做什么的

这个 react 生命周期钩子函数是来解决这个问题：
在更新数据的时候用 setState 修改整个数据，数据变了之后，遍历的时候所有内容都要被重新渲染，数据量少还好，数据量大就会严重影响性能

解决办法：

1.shouldcomponentupdate 在渲染前进行判断组件是否更新，更新了再渲染
2.purecomponent（纯组件）省去了虚拟 dom 生成和对比的过程 在类组件中使用
3.react.memo() 类似于纯组件 在无状态组件中使用

**13.react diff 原理**

它是基于三个策略：

tree diff web UI 中 dom 节点跨层级的移动操作特别少，可以忽略不计
component diff 拥有相同类的两个组件将会生成相似的树形结构，拥有不同类的两个组件会生成不同的树形结构
element diff 对于同一层级的一组子节点，他们可以通过唯一的 id 进行区分

**14.何为受控组件**

React 负责渲染表单的组件，值是来自于 state 控制的，输入表单元素称为受控组件

**15.调用 super(props) 的目的是什么**

Super（）调用父类的构造方法，有 super，组件才有自己的 this，在组件全局中都可以使用 this，如果只是 constructor 而不执行 super，之后的 this 都是错的，super 继承父组件的 this

## 16.React 中构建组件的方式

**自定义组件**：函数组件或者无状态组件 组件首字母大写
**类组件**：一个类组件必须实现一个 render 方法，这个方法必须返回一个 jsx 元素，要用一个外层的元素把所有内容包裹起来

## 17.简述 flux 思想

Flux 的最大特点，就是数据的"单

向流动"。 1.用户访问 View

2. View 发出用户的 Action

3. Dispatcher 收到 Action，要求 Store

进行相应的更新 4.Store 更新后，发出一

个"change"事件

5. View 收到"change"事件后，更新页面

## 18.React 项目用过什么脚手架？Mern? Yeoman?

Mern：MERN 是脚手架的工具，它可以很容易地使用 Mongo, Express, React and NodeJS 生成同构 JS 应用。它最大限度地减少安装时间，并得到您使用的成熟技术来加速开发。

## 19.应该在 React 组件的何处发起 Ajax 请求？

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

## 20.何为高阶组件(higher order component)？

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

# 小程序相关

## 1.小程序的优势

无需下载安装，直接使用，运行速度快，项目搭建迅速，短小精悍，每个 app 源代码不超过 2mb

## 2.小程序的页面构成（4 个文件）

Index.js index.json index.wxml index.wxss

## 3.小程序的生命周期

Onload onready onshow onhide onunload

Onpulldownrefresh onreachbottom onshareappmessage

## 4.小程序如何请求数据

用 request

## 5.如何提高小程序的首屏加载时间

提前请求：异步数据数据请求不需要等待页面渲染完成

利用缓存：利用 storage API 对异步请求数据进行缓存，二次启动时先利用缓存数据渲染页面，再进行后台更新

避免白屏：先展示页面骨架和基础内容

及时反馈：及时地对需要用户等待的交互操作给出反馈，避免用户以为小程序没有响应

性能优化：避免不当使用 setdata 和 onpagescroll

## 6.请简述你经常使用的小程序的组件

View icon text image swiper navigator input button map

## 7.Wxss 与 css 的区别请简述

Wxss 新增了尺寸单位 rpx

提供了全局样式和局部样式

Wxss 仅支持部分 css 选择器 id’ class 元素等

## 8.小程序如何实现响应式

Rpx

## 9.怎么优化小程序

提高页面加载速度

用户行为预测

减少默认 data 的大小

组件化方案

自主获知自己的服务器

## 10.小程序如何显示用户头像与用户名

传统接口 wx.getuserinfo 目前可以用，需要用户授权，使用时会有官方发提示，这个方法需要升级

最新方法：open-data 标签，使用这个标签可以不用用户授权直接获取头像和用户名，
可以在 button 中将 opendata 作为属性写进去，写个点击事件就直接获取到了

## 11.请谈谈小程序的双向绑定和 vue 的异同？

Vue双向绑定是通过数据拦截和观察者模式，通过this.value获取值，小程序是通过触发表单元素绑定的方法，在方法中用 this.setData（{key:value}）来取值

**12.**小程序中传参是怎么传的

**13.**和 **vue** 类比介绍

**14.**说一下微信小程序的适配问题

**15.**小程序页面间有哪些传递数据的方法？

**16.**你是怎么封装微信小程序的数据请求的

**17.**说一下微信小程序的适配问题

**18.**小程序跳转页面的方式

**19.**微信小程序如何跳转到其他小程序

**20.**小程序加载过慢的解决方式

# 其他

**1.Typescript** 是什么 请简述？

**2.Typescript** 与 **javascript** 的优势？

**3.Webpack** 与 **gulp** 区别

Gulp 是一种能够优化前端开发流程的工具，webpack 是一种模块化

的解决方案 （grunt）

**4.请简述 webpack 中的 loaders 与 plugin 的区别**

什么是 loaders，loaders 是文件加载器，能够加载资源文件，并对这些文件进行处理，例如，编译，压缩等，最终一起打包到指定文件中。

什么是 plugin，在 webpack 运行的生命周期会有许多事件，plugin 可以监听这些事件

区别：加载器是用来加载文件的，webpack 本身只能加载 js 文件(内置 babel-loader)，加载其他文件就需要安装别的 loader，比如：css-loader　file-loader

Plugin 是扩展 webpack 功能的，通过 plugin　，webpack 可以实现 loader 不能完成的复杂功能

**5.怎么提升页面性能？性能优化有哪些？**

**6.Node 使用来做什么的**

能够在服务器端运行 JavaScript

Webpack：入口，出口，加载器，插件

**7.说一下 webpack 的打包原理**

Webpack 是把项目当做一个整体，通过给定一个主文件，webpack 将从这个主文件开始找到项目中所有依赖的文件，使用 loaders 类处理，最后打包成一个或者多个浏览器可识别的 js 文件

**8.Commonjs ES6 模块区别？**

common 模块是拷贝，可以修改值，es6 模块是引用，只读状态，不能修改值

commonjs 模块是运行时加载，es6 模块是编译时输出接口

**9.Git 如何使用/常用指令有哪些**

**10.你们后台用的是什么技术**

**11.你的项目比较小为什么还是用 vue 全家桶**

**12.请简述你在项目中使用的 ui 框架**

**13.什么是 cors**

**14.说一下对 websocket 的理解**

Websocket 是一种双向通信协议，在建立连接后，websocket 服务器和浏览器都能主动向对方发送或者接收数据，websocket 需要类似于 tcp 的客户端和服务器通过握手连接，连接成功后才能互相通信

**15.后台传递过来的数据是那些**

**16.谈谈 Ajax，fetch，axios 的区别**

# 企业中的项目流程

## 1.WEB 前端项目开发流程

**项目需求分析**

这个环节是由项目经理完成，项目经理首先和客户进行交流，了解客户的需求，然后分析项目的可行性，如果项目可以被实现，项目经理写出项目需求文档交给设计师完成后续的开发。

**页面设计/项目选型**

这个环节主要是 UI 设计师参与，UI 设计师根据产品需求分析文档，对产品的整体美术风格、交互设计、界面结构、操作流程等做出设计。负责项目中各种交互界面、图标、LOGO、按钮等相关元素的设计与

制作。并且确定使用技术

**编码**

这个部分由程序员来实现。(程序员分为 WEB 前端开发工程师和后台开发工程师。前端开发人员主要做我们可以在网页上看的见的页面，后台就做一些我们看不见的管理系统以及功能的实现。）程序员根据 UI 设计师的设计，用编码来完成整个项目的各个功能。

**测试**

这部分由程序测试员来完成。程序测试员主要就是测试寻找程序还存在的 bug，一般来说刚编码完成的程序都是存在问题的，就需要测试人员反复不断的测试并将存在问题的测试结果交给编码人员进行 bug 的修复。等到几乎所有 bug 修复完成，这个项目差不多就可以上线了。

**维护**

程序的维护是整个项目的最后一个阶段，但也是耗时最多，成本最高最高的的一个阶段。程序的维护包括程序上线后后续 bug 的修复和程序版本的更新。

**更换接口域名**

就是在开发的时候调用的后台接口是后台测试的接口　项目上线后

要把请求的接口替换成上线的域名

**经常使用的工具**

代码管理平台：github 码云

需求发布平台：钉钉任务，禅道

Ui 交互平台：蓝湖

产品原型工具：axure

企业邮箱：阿里 腾讯企业邮箱

后台语言：java php python（西安不多）

## 2 大公司和小公司开发的区别

大型外包公司更加流程化，人员多，沟通少，项目交付后不需要自己维护，采用瀑布开发模式（以文档为主）

小型公司：人少 需求经常改变 沟通方便 采用敏捷开发（快速推出 v1 版本 ，之后迭代）

2.前后台分离怎么测试?

# 奇葩问题

**1.你们后端开发用的什么?**

**2.移动端如何刷新页面?**

**3.项目初始化构建流程**

**4.项目中自己觉得骄傲的地方?**

**5.说说自己的缺点**

**6.热部署是什么?**

**7.用户有多少**

**8 怎么调用接口（是怎么跟后台沟通的）**

**9.单元格测试是怎么做的**

**10.开发环境，测试环境，上线环境的环境变量你们在开发中是如何处理的**

---

本文收集整理于 CSDN 博主「赫兹/Herzz」的原创文章

原文链接:

https://blog.csdn.net/HanXiaoXi_yeal/article/details/112306277