5月最新大厂前端高频核心面试题

一、HTML5 面试题（附答案）

1. HTML5 有哪些新特性，移除了那些元素？如何区分 HTML 和 HTML5?

新增加了图像、位置、存储、多任务等功能。

新增元素：

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

位置 API：Geolocation

表单控件：calendar date time email url search

新的技术：

web worker(web worker 是运行在后台的 JavaScript，独立于其他脚本，不会影响页面的性能。您可以继续做任何愿意做的事情：点击、选取内容等等，而此时 web worker 在后台运行)

web socket

拖放 API：drag、drop

移除的元素：

纯表现的元素：basefont big center font s strike tt u

性能较差元素：frame frameset noframes

区分：

DOCTYPE 声明的方式是区分重要因素

根据新增加的结构、功能来区分

2. 如何处理 HTML5 新标签的浏览器兼容问题?

IE6/IE7/IE8 支持通过 `document` 方法产生的标签，利用这一特性让这些浏览器支持 HTML5 新标签；

使用静态资源的 `html5shiv` 包：

1. 在 `<head>` 中调用以下代码：

<!--[if lt IE9]>

```
<script
src="http://cdn.static.runoob.com/libs/html5shiv/3.7/html5shiv.min.js"
></script>
<![endif]-->
```

2.载入后，初始化新标签的 css：

```
header, section, footer, aside, nav, main, article, figure { display: block; }
```

3. HTML5 的文件离线储存怎么使用，工作原理是什么?

在线情况下，浏览器发现 HTML 头部有 manifest 属性，它会请求 manifest 文件，如果是第一次访问，那么浏览器就会根据 manifest 文件的内容下载相应的资源，并进行离线存储。如果已经访问过并且资源已经离线存储了，那么浏览器就会使用离线的资源加载页面。然后浏览器会对比新的 manifest 文件与旧的 manifest 文件，如果文件没有发生改变，就不会做任何操作，如果文件改变了，那么就会重新下载文件中的资源，并且进行离线存储。

在页面头部加入 manifest 属性

```
1<html manifest='cache.manifest'>
```

在 cache.manifest 文件中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

二、HTML 高级面试题（附答案）

1. 请用 html 知识解决 seo 优化问题?

meta 标签，这个是重中之重

```
<title>html 对 seo 的优化</title>

<meta name="title" content="html 对 SEO 的优化">/*不推荐用这个*/

<meta name="keywords" content="SEO,爬虫，搜索引擎、百度、html 优化">

<meta name="description" content="通过 html 标签及属性的使用提高网站被爬虫爬取的几率，使用户百度时网站尽量排在前面，提高用户的点击率">

logo，给 logo 图片添加 h1 标签、a 链接连接到首页以及 alt


<h1>

  // 这个 href 应该是要写线上的首页地址，比项目目录地址要好

  <a href="https://xxx">

      <img src="images/logo.png"  alt="html 对 seo 的优化"/>
```


</h1>

img 标签，img 标签增加 alt 属性

a 标签

a 标签增加 title 属性，不可以有 href="#" 这种空指向写法，另外大量的 title 感觉体验也不是很好，不需要的地方可以不用。

h1~h6 标签

h1 要分配给网站名称或给带 alt 标签的 logo 使用（这个前面也提到了），用以强调网站名称。

h2 标签用来定义站点副标题。如果没有副标题，h2 标签最好也空着，以备不时之需。

h3 标签用来定义导航栏目名称。

h4 标签用来定义文章列表标题，但大多数内容系统，文章列表输出用 UL 标签，所以 h4 可能就派不上用场，这里只是以此类推。浏览器会自动地在标题的前后添加空行。请确保将 HTML heading 标签只用于标题。不要仅仅是为了产生粗体或大号文本而使用标题因为搜索引擎使用标题为你的网页的结构和内容标志索引。

添加 robots.txt ，搭建网站与搜索引擎对话的桥梁

在项目根目录添加 robots.txt 文件，robots.txt 文件可以告诉搜索引擎哪些是重点，哪些又是可以忽略的，节约搜索引擎蜘蛛抓取网页的时间，也在一定程度上节省了服务器资源。

页面结构清晰

使用语义化标签比如 header、footer、content、section，js、css 使用外部文件。

增加外部链接

即是对方没有和你链接，你也是可以链接别人的，大概搜索引擎的算法体现了互联网“分享”的精神吧，通过外链网站的活跃度蹭点 seo 度。

前后端分离（vue、ajax）、flash 不利于 seo

2. 常用浏览器有哪些，内核都是什么？

常用浏览器有 IE 火狐(firefox) chrome safari 360 搜狗 等

内核：

IE 的是 Trident

火狐的是 Gecko

chrome 和 safari 用的是 Webkit

360 和搜狗这些分极速模式和兼容模式，极速模式用的 Webkit 的内核，兼容模式用的 Trident 内核。

3. 介绍一下你对浏览器内核的理解？

浏览器内核主要分成两部分：渲染引擎(Layout Engine 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。

JS 引擎：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。

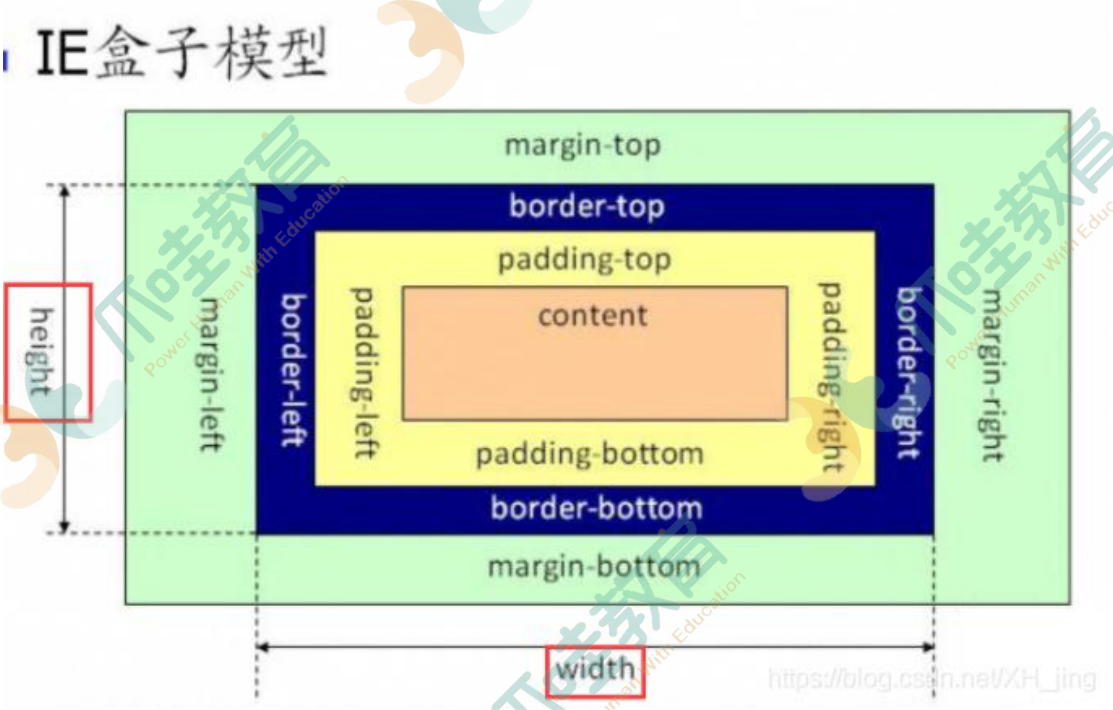
4. 浏览器标准模式和怪异模式之间的区别是什么?

在标准模式页面按照 HTML、CSS 的定义渲染，而在怪异模式就是浏览器为了兼容很早之前针对旧版本浏览器设计，并未严格遵循 W3C 标准而产生的一种页面渲染模式。浏览器基于页面中文件类型描述的存在以决定采用哪种渲染模式，如果存在一个完整的 DOCTYPE 则浏览器将会采用标准模式，如果缺失就会采用怪异模式。

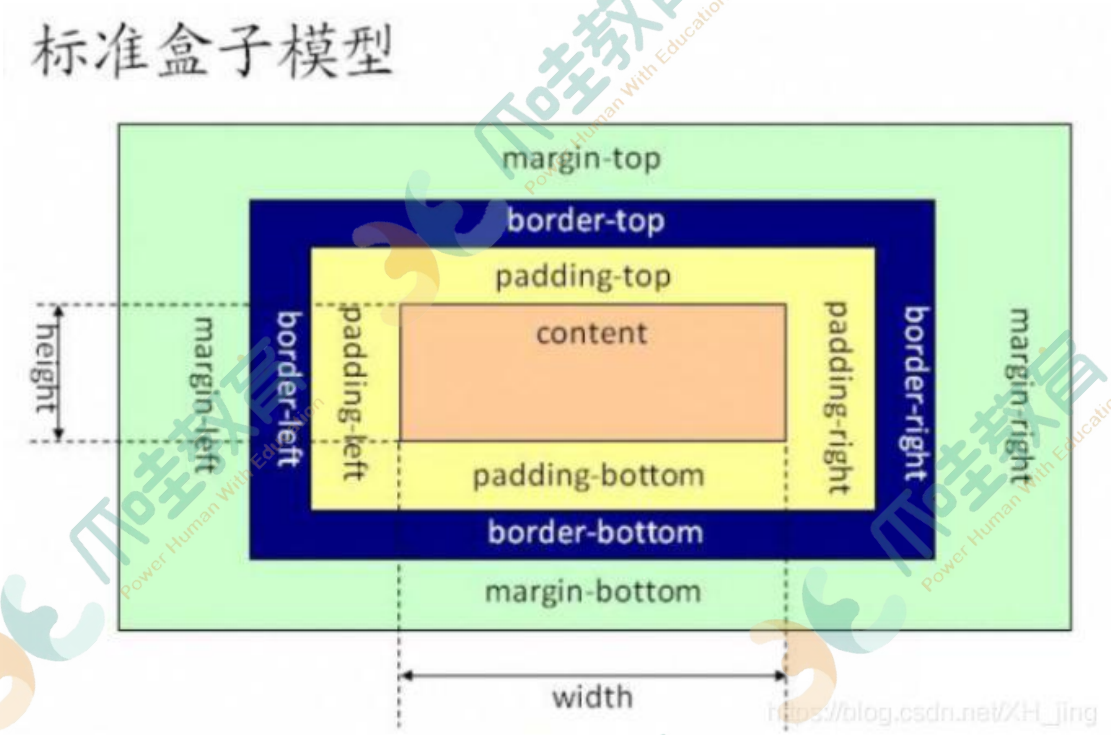
区别：

盒模型

在怪异模式下，盒模型为 IE 盒模型：



W3C 标准的盒模型：



图片元素的垂直对齐方式

对于 inline 元素和 table-cell 元素，标准模式下 vertical-align 属性默认取值为 baseline，在怪异模式下，table 单元格中的图片的 vertical-align 属性默认取值为 bottom，因此在图片底部会有及像素的空间。(图片底部空白，通过设置 vertical-align：center 来解决)

<table>元素中的字体

CSS 中，对于 font 的属性都是可以继承的，怪异模式下，对于 table 元素，字体的某些元素将不会从 body 等其他封装元素中继承得到，特别是 font-size 属性。

内联元素的尺寸

标准模式下，non-replaced inline 元素无法自定义大小，怪异模式下，定义这些元素的 width，height 属性可以影响这些元素显示的尺寸。

元素的百分比高度

CSS 中对于元素的百分比高度规定如下：百分比为元素包含块的高度，不可为负值，如果包含块的高度没有显示给出，该值等同于 auto，所以百分比的高度必须在父元素有高度声明的情况下使用。

当一个元素使用百分比高度时，标准模式下，高度取决于内容变化，怪异模式下，百分比高度被正确应用。

元素溢出的处理

标准模式下，overflow 取默认值 visible；

在怪异模式下，该溢出会被当做扩展 box 来对待，即元素的大小由其内容决定，溢出不会裁减，元素框自动调整，包含溢出内容。

5. 渐进增强（progressive enhancement）和优雅降级（graceful degradation）的区别?

渐进增强：先保证低版本浏览器的基本功能，再去兼容高版本浏览器效果和交互。

优雅降级：先保证高版本浏览器的效果和交互等，再去兼容低版本的浏览器。

6. 无样式内容闪烁（FOUC）Flash of Unstyle Content?

什么是 FOUC(文档样式短暂失效)?

如果使用@import 方法对 CSS 进行导入，会导致某些页面在 Windows 下的 Internet Explorer 出现一些奇怪的现象：以无样式显示页面内容的瞬间闪烁，这种现象称之为文档样式短暂失效(Flash of Unstyled Content)，简称为 FOUC。

原因：

使用@import 方法导入样式表

将样式表放在页面底部

有几个样式表，放在 html 结构的不同位置。

其实原理很清楚：当样式表晚于结构性 html 加载，当加载到此样式表时，页面将停止之前的渲染。此样式表被下载和解析后，将重新渲染页面，也就出现了短暂的花屏现象。

解决方法：

使用 link 标签加载 CSS 样式文件。因为 link 是顺序加载的，这样页面会等到 CSS 下载完之后再下载 HTML 文件，这样先布局好，就不会出现 FOUC 问题。

7. 为什么通常将 css 的<link>放置在<head></head>之间，而将 js 的<script>放置在</body>之前?有哪些例外吗?

浏览器在处理 HTML 页面渲染和 JavaScript 脚本执行的时候是单一进程的，所以在当浏览器在渲染HTML遇到了<script>标签会先去执行标签内的代码(如果是使用src属性加载的外链文件,则先下载再执行)，在这个过程中，页面渲染和交互都会被阻塞。所以将<script>放在</body>之前，当页面渲染完成再去执行<script>。

一般希望 DOM 还没加载必须需要先加载的 js 会放置在<head>中，有些加了 defer、async 的<script>也会放在<head>中。

8. <script>标签中 defer 和 async 属性的区别?

async：与后续元素渲染异步执行，乱序执行，若 js 文件之间存在依赖关系，容易产生错误，只适用于完全没有依赖的文件，文档解析过程中异步下载，下载完成之后立即执行。

defer：（H5 规范中，defer 是有序执行的，但实际不一定是有序执行的）与后续渲染异步执行，延迟到界面文档解析完成之后执行，即为立即下载，延迟执行。所有执行均在 DOMContentLoaded 事件触发之前完成。

async 和 defer 异步加载，async 下载完立即执行，defer 待界面文档解析完成之后执行

不带属性：加载到<script>立即下载并执行，阻塞后续渲染的执行。

最佳方案：外部引用文件放在/body 之前执行

9. data- 属性的作用是什么？

data- 为 H5 新增的为前端开发者提供自定义的属性，这些属性集可以通过对象的 dataset 属性获取，不支持该属性的浏览器可以通过 getAttribute 方法获取 ；

需要注意的是：data- 之后的以连字符分割的多个单词组成的属性，获取的时候使用驼峰风格。 所有主流浏览器都支持 data-* 属性。

即：当没有合适的属性和元素时，自定义的 data 属性是能够存储页面或 App 的私有的自定义数据。

10. 简述一下你对 HTML5 语义化的理解？

很多不错的公司都会问语义化的问题，这里有篇文章讲得很好，推荐给大家：html5 语义化标签

去掉或丢失样式的时候能够让页面呈现出清晰的结构。

有利于 SEO 和搜索引擎建立良好沟通，有助于爬虫抓取更多的信息，爬虫依赖于标签来确定上下文和各个关键字的权重。

方便其它设备解析。

便于团队开发和维护，语义化根据可读性。

11. Cookies，sessionStorage 和 localStorage 的区别?

共同点：都是保存在浏览器端，且是同源的。

区别：

存储位置不同：cookies 是为了标识用户身份而存储在用户本地终端上的数据，始终在同源 http 请求中携带，即 cookies 在浏览器和服务器间来回传递，而 sessionStorage 和 localStorage 不会自动把数据发给服务器，仅在本地保存。

存储大小的限制不同：cookie 保存的数据很小，不能超过 4k，而 sessionStorage 和 localStorage 保存的数据大，可达到 5M。

数据的有效期不同：cookie 在设置的 cookie 过期时间之前一直有效，即使窗口或者浏览器关闭。sessionStorage 仅在浏览器窗口关闭之前有效。localStorage 始终有效，窗口和浏览器关闭也一直保存，用作长久数据保存。

作用域不同：cookie 在所有的同源窗口都是共享；sessionStorage 不在不同的浏览器共享，即使同一页面；localStorage 在所有同源窗口都是共享。

12. iframe 框架有那些优缺点?

优点：

iframe 能够原封不动的把嵌入的网页展现出来；

如果有多个网页引用 iframe，那么你只需要修改 iframe 的内容，就可以实现调用的每一个页面内容的更改，方便快捷；

网页如果为了统一风格，头部和版本都是一样的，就可以写成一个页面，用 iframe 来嵌套，可以增加代码的可重用；

如果遇到加载缓慢的第三方内容如图标和广告，这些问题可以由 iframe 来解决。

缺点：

搜索引擎的爬虫程序无法解读这种页面；

框架结构中出现各种滚动条；

使用框架结构时，保证设置正确的导航链接；

iframe 页面会增加服务器的 http 请求。

13. label 的作用是什么？是怎么用的?

label 标签用来定义表单控件间的关系，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 中有两个属性是非常有用的，for 和 accesskey。

for 属性功能：表示 label 标签要绑定的 HTML 元素，你点击这个标签的时候，所绑定的元素将获取焦点。

<Label for="name">姓名</Label><input ID="name" type="text">

accesskey 属性功能：表示访问 label 标签所绑定的元素的热键，当您按下热键，所绑定的元素将获取焦点。

<Label for="name" accesskey = "N"> 姓 名 </Label><input ID="name" type="text">

14. HTML5 的 form 如何关闭自动完成功能?

HTML 的输入框可以拥有自动完成的功能，当你往输入框输入内容的时候，浏览器会从你以前的同名输入框的历史记录中查找出类似的内容并列在输入框下面，这样就不用全部输入进去了，直接选择列表中的项目就可以了。但有时候我们希望关闭输入框的自动完成功能，例如当用户输入内容的时候，我们希望使用 AJAX 技术从数据库搜索并列举而不是在用户的历史记录中搜索。

方法：

在 IE 的 internet 选项菜单中里的自动完成里面设置

设置 form 输入框的 autocomplete 为 on 或者 off 来来开启输入框的自动完成功能

15. 如何实现浏览器内多个标签页之间的通信?

WebSocket、SharedWorker

也可以调用 localstorge、cookies 等本地存储方式。 localstorge 在另一个浏览上下文里被添加、修改或删除时，它都会触发一个事件，我们通过监听事件，控制它的值来进行页面信息通信。

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

16. webSocket 如何兼容低浏览器?

Adobe Flash Socket ActiveX HTMLFile (IE) 基于 multipart 编码发送 XHR 基于长轮询的 XHR

引用 WebSocket.js 这个文件来兼容低版本浏览器。

17. 页面可见性（Page Visibility）API 可以有哪些用途?

通过 visibility state 的值得检测页面当前是否可见，以及打开网页的时间；

在页面被切换到其他后台进程时，自动暂停音乐或视频的播放。

18. 如何在页面上实现一个圆形的可点击区域?

map + area 或者 svg

border-radius

纯 js 实现，一个点不在圆上的算法

19. 实现不使用 border 画出 1px 高的线，在不同浏览器的 Quirks mode 和 CSS Compat 模式下都能保持同一效果

```
<div style="height:1px;overflow:hidden;background:red"></div>
```

20. 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程序;

可以防止恶意破解密码、刷票、论坛灌水;

21. 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，只表示标题；h1 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 标明重点内容，语气加强含义；b 是无意义的视觉表示;

em 表示强调文本；i 是斜体，是无意义的视觉表示;

视觉样式标签：b i u s

语义样式标签：strong em ins del code

22. 写了 2 个 <a> 标签，两个标签之间有空格的情况遇到过吗?

遇到过这种情况，一般将两个 <a> 标签换行写的时候会出现这种情况

```
<a href="https://www.baidu.com">百度 1</a>

<a href="https://www.baidu.com">百度 2</a>
```

解决方法：将两个 <a> 标签 合在一行就可以了。

```
<a href="https://www.baidu.com">百度 1</a><a href="https://www.baidu.com">百度 2</a>
```

23. 什么是渐进式渲染?

渐进式渲染是用于提高网页性能（尤其是提高用户感知的加载速度），以尽快呈现页面的技术。

在以前互联网带宽较小的时期，这种技术更为普遍。如今，移动终端的盛行，而移动网络往往不稳定，渐进式渲染在现代前端开发中仍然有用武之地。

比如：

（1）图片懒加载——页面上的图片不会一次性全部加载。当用户滚动页面到图片部分时，JavaScript 将加载并显示图像。

（2）确定显示内容的优先级（分层次渲染）——为了尽快将页面呈现给用户，页面只包含基本的最少量的 CSS、脚本和内容，然后可以使用延迟加载脚本或监听 DOMContentLoaded/load 事件加载其他资源和内容。

（3）异步加载 HTML 片段——当页面通过后台渲染时，把 HTML 拆分，通过异步请求，分块发送给浏览器。

三、CSS3 面试题（附答案）

1. CSS3 有哪些新特性?

RGBA 和 透明度

background-image background-origin(content-box/padding-box/border-box) background-size background-repeat

word-wrap（对长的不可分割单词换行）word-wrap：break-word

文字阴影：text-shadow： 5px 5px 5px #FF0000;（水平阴影，垂直阴影，模糊距离，阴影颜色）

font-face 属性：定义自己的字体

圆角（边框半径）：border-radius 属性用于创建圆角

边框图片：border-image: url(border.png) 30 30 round

盒阴影：box-shadow: 10px 10px 5px #888888

媒体查询：定义两套 css，当浏览器的尺寸变化时会采用不同的属性

2. 解释一下 Flexbox (弹性盒布局模型)？及适用场景？

flex 布局是 CSS3 新增的一种布局方式，我们可以通过将一个元素的 display: flex 使他成为一个 flex 容器。任何一个容器都可以指定为 flex 布局。行内元素也可使用 flex 布局。

一个容器默认有两条轴，一个是水平的主轴，一个是与主轴垂直的交叉轴。

属性：

flex-direction 定义主轴的方向；

flex-wrap 定义是否换行；

flex-flow 上述 2 个属性的简写；

justify-content 定义项目在主轴上的对齐方式；

align-items 定义项目在交叉轴上如何对齐；

align-content 定义多根轴线的对齐方式

3. CSS3 新增伪类有那些？

p:first-of-type 选择属于其父元素的首个元素

p:last-of-type 选择属于其父元素的最后元素

p:only-of-type 选择属于其父元素唯一的元素

p:only-child 选择属于其父元素的唯一子元素

p:nth-child(2) 选择属于其父元素的第二个子元素

:enabled :disabled 表单控件的禁用状态。

:checked 单选框或复选框被选中。

四、CSS 高级面试题（附答案）

1. 浏览器兼容性有哪些?

TODO

2. 使用 base64 编码的优缺点?

base64 编码是一种图片处理格式，通过特定的算法将图片编码成一长串字符串，在页面上显示时可用该字符串来代替图片的 url 属性

使用 base64 的优点:

可以减少该图片的 HTTP 请求

使用 base64 的缺点:

1.根据 base64 的编码原理，编码后的大小会比源文件大小大 1/3，如果把大图片编码到 html/css 中，

不仅会造成文件体积增加，影响文件的加载速度，还会增加浏览器对 html 或 css 文件解析渲染的时间。

2.使用 base64 无法直接缓存，要缓存只能缓存包含 base64 的文件，比如 HTML 或 CSS，这相比于直接缓存图片的效果要差很多。

3.IE8 以前的浏览器不支持，一般一些网站的小图标可以使用 base64 图片引入。

3. 为什么要清除浮动？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。

清除浮动的方式：

见笔记 TODO

4. CSS 优化，提高性能的方法有哪些？

加载性能：

CSS 压缩：将写好的 CSS 进行打包压缩，可以减少很多的体积。

CSS 单一样式：当需要下边距和左边距的时候，很多时候选择：margin: 0 0; 比 margin-top: 0; margin-bottom: 0; 执行的效率更高。

选择器性能：

关键选择器。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）。

5. 如何实现一个 div 的上下垂直居中？

TODO

6. CSS 里的 visibility 属性有个 collapse 属性值？在不同浏览器下有什么区别？

当一个元素的 visibility 属性被设置成 collapse 值后，对于一般的元素，它的表现跟 hidden 是一样的

Chrome 中，使用 collapse 值和使用 hidden 没有区别；

firefox，opera 和 IE，使用 collapse 值和使用 display：none 没有什么区别。

7. display:none 与 visibility：hidden 的区别?

display：none 不显示对应的元素，在文档布局中不再分配空间（回流+重绘）

visibility：hidden 隐藏对应元素，在文档布局中仍保留原来的空间（重绘）

8. position 跟 display、overflow、float 这些特性相互叠加后会怎么样?

display 属性规定元素应该生成的框的类型；

position 属性规定元素的定位类型；

float 属性是一种布局方式，定义元素在哪个方向浮动。

类似于优先级机制：position：absolute/fixed 优先级最高，有他们在时，float 不起作用，display 值需要调整。float 或者 absolute 定位的元素，只能是块元素或表格。

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

BFC 规定了内部的 Block Box 如何布局。

具体深入理解请移步至：对 BFC 的深层理解

10. 上下 margin 重合的问题?

在重合元素外包裹一层容器，通过改变此 div 的属性使两个盒子分属于两个不同的 BFC，以此来阻止 margin 重叠。

11. 移动端的布局用过媒体查询吗?

通过媒体查询可以为不同大小和尺寸的媒体定义不同的 css，适应相应的设备的显示。

<head> 里边引入： <link rel=" stylesheet" type=" text/css" href=" xxx.css" media=" only screen and (max-device-width:480px)" >

CSS 中定义：@media only screen and (max-device-width:480px) { /* css 样式 */ }

12. CSS 优化、提高性能的方法有哪些?

避免过度约束

避免后代选择符

避免链式选择符

使用紧凑的语法

避免不必要的命名空间

避免不必要的重复

最好使用表示语义的名字。一个好的类名应该是描述他是什么而不是像什么

避免 !important，可以选择其他选择器

尽可能的精简规则，你可以合并不同类里的重复规则

13. 浏览器是怎样解析 CSS 选择器的?

CSS 选择器的解析是从右向左解析的。

若从左向右的匹配，发现不符合规则，需要进行回溯，会损失很多性能。

若从右向左匹配，先找到所有的最右节点，对于每一个节点，向上寻找其父节点直到找到根元素或满足条件的匹配规则，则结束这个分支的遍历。

两种匹配规则的性能差别很大，是因为从右向左的匹配在第一步就筛选掉了大量的不符合条件的最右节点（叶子节点），而从左向右的匹配规则的性能都浪费在了失败的查找上面。

而在 CSS 解析完毕后，需要将解析的结果与 DOM Tree 的内容一起进行分析建立一棵 Render Tree，最终用来进行绘图。在建立 Render Tree 时（WebKit 中的 Attachment 过程），浏览器就要为每个 DOM Tree 中的元素根据 CSS 的解析结果（Style Rules）来确定生成怎样的 Render Tree。

14. 在网页中的应该使用奇数还是偶数的字体？为什么呢？

尽量使用偶数字体。偶数字号相对更容易和 web 设计的其他部分构成比例关系。Windows 自带的点阵宋体（中易宋体）从 Vista 开始只提供 12、14、16 px 这三个大小的点阵，而 13、15、17 px 时用的是小一号的点。（即每个字占的空间大了 1 px，但点阵没变），于是略显稀疏。

15. 全屏滚动的原理是什么？用到了 CSS 的哪些属性？

原理：有点类似于轮播，整体的元素一直排列下去，假设有 5 个需要展示的全屏页面，那么高度是 500%，只是展示 100%，剩下的可以通过 transform 进行 y 轴定位，也可以通过 margin-top 实现。

属性：overflow：hidden；transition：all 1000ms ease；

16. 什么是响应式设计？响应式设计的基本原理是什么？如何兼容低版本的 IE?

响应式网站设计(Responsive Web design)是一个网站能够兼容多个终端，而不是为每一个终端做一个特定的版本。

基本原理是通过媒体查询 @media 检测不同的设备屏幕尺寸做处理。

兼容低版本 IE 可以使用 JS 辅助一下来解决。

17. 视差滚动效果？

视差滚动（Parallax Scrolling）通过在网页向下滚动的时候，控制背景的移动速度比前景的移动速度慢来创建出令人惊叹的 3D 效果。

实现方式：

CSS3 实现

优点：开发时间短、性能和开发效率比较好；

缺点是不能兼容到低版本的浏览器

jQuery 实现

通过控制不同层滚动速度，计算每一层的时间，控制滚动效果。

优点：能兼容到各个版本的，效果可控性好

缺点：开发起来对制作者要求高

插件实现方式

例如：parallax-scrolling，兼容性比较好

18. ::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。

::before 就是以一个子元素的存在，定义在元素主体内容之前的一个伪元素。并不存在于 dom 之中，只存在于页面之中。

:before 和 :after 这两个伪元素，是在 CSS2.1 里新出现的。起初，伪元素的前缀使用的是单冒号语法，但随着 Web 的进化，在 CSS3 的规范里，伪元素的语法被修改成使用双冒号，成为 ::before ::after。

19. 怎么让 Chrome 支持小于 12px 的文字?

使用 transform:scale() 进行缩放即可实现。

```
p{
        font-size:10px;
    -webkit-transform:scale(0.8); //0.8 是缩放比例
}
```

20. 让页面里的字体变清晰，变细用 CSS 怎么做?

-webkit-font-smoothing 在 window 系统下没有起作用，但是在 IOS 设备上起作用

-webkit-font-smoothing：antialiased 是最佳的，灰度平滑。

21. position:fixed;在 android 下无效怎么处理?

< meta name="viewport"

content="width=device-width, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0, user-scalable=no"

/>

22. 如果需要手动写动画，你认为最小时间间隔是多久，为什么?

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

23. 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。

解决方法：

可以将 <li> 代码全部写在一排

浮动 li 中 float：left

在 ul 中用 font-size：0（谷歌不支持）；可以使用 letter-space：-3px

24. display:inline-block 什么时候会显示间隙?

有空格时候会有间隙——解决：移除空格

margin 正值的时候——解决：margin 使用负值

使用 font-size 时候——解决：font-size:0、letter-spacing、word-spacing

25. png、jpg、gif 这些图片格式解释一下，分别什么时候用。有没有了解过 webp?

png 是便携式网络图片（Portable Network Graphics）是一种无损数据压缩位图文件格式。优点是：压缩比高，色彩好。 大多数地方都可以用。

jpg 是一种针对相片使用的一种失真压缩方法，是一种破坏性的压缩，在色调及颜色平滑变化做的不错。在 www 上，被用来储存和传输照片的格式。

gif 是一种位图文件格式，以 8 位色重现真色彩的图像。可以实现动画效果。

webp 格式是谷歌在 2010 年推出的图片格式，压缩率只有 jpg 的 2/3，大小比 png 小了 45%。缺点是压缩的时间更久了，兼容性不好，目前谷歌和 opera 支持。

26. style 标签写在 body 后与 body 前有什么区别?

页面加载自上而下 当然是先加载样式。

写在 body 标签后由于浏览器以逐行方式对 HTML 文档进行解析，当解析到写在尾部的样式表（外联或写在 style 标签）会导致浏览器停止之前的渲染，等待加载且解析样

式表完成之后重新渲染，在 windows 的 IE 下可能会出现 FOUC 现象（即样式失效导致的页面闪烁问题）

27. CSS 属性 overflow 属性定义溢出元素内容区的内容会如何处理?

参数是 scroll 时候，必会出现滚动条。

参数是 auto 时候，子元素内容大于父元素时出现滚动条。

参数是 visible 时候，溢出的内容出现在父元素之外。

参数是 hidden 时候，溢出隐藏。

28. 阐述一下 CSS Sprites（雪碧图）

将一个页面涉及到的所有图片都包含到一张大图中去，然后利用 CSS 的 background-image，background- repeat，background-position 的组合进行背景定位。

利用 CSS Sprites 能很好地减少网页的 http 请求，从而大大的提高页面的性能；CSS Sprites 能减少图片的字节。

29. Sass、Less 是什么？大家为什么要使用他们?

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

Less 是一种动态样式语言。将 CSS 赋予了动态语言的特性，如变量，继承，运算，函数。Less 即可以在客户端上运行（支持 IE6+,Webkit,Firefox），也可以在服务端运行（借助 Node.js）。

Sass 变量必须是 $ 开始，而 Less 变量必须使用 @ 符号开始。

为什么要使用它们?

结构清晰，便于扩展。

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

可以轻松实现多重继承。

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

五、微信小程序

1. 简单描述下微信小程序的相关文件类型?

微信小程序项目结构主要有四个文件类型：

wxml 模板文件，是框架设计的一套标签语言，结合基础组件、事件系统、可以构建出页面的结构；

wxss 样式文件，是一套样式语言，用于描述 WXML 的组件样式；

js 脚本逻辑文件，逻辑处理网络请求；

json 配置文件，小程序设置，如页面注册，页面标题及 tabBar；

app.json 整个小程序的全局配置，包括：

pages: [所有页面路径]

网络设置（网络超时时间）

界面表现（页面注册）

window: {背景色、导航样式、默认标题}

底部 tab 等

app.js 监听并处理小程序的生命周期函数、声明全局变量；

app.wxss 全局配置的样式文件。

2. 请谈谈 wxml 与标准的 html 的异同?

都是用来描述页面的结构；

都由标签、属性等构成；

标签名字不一样，且小程序标签更少，单一标签更多；

多了一些 wx:if 这样的属性以及 {{ }} 这样的表达式；

WXML 仅能在微信小程序开发者工具中预览，而 HTML 可以在浏览器内预览；

组件封装不同， WXML 对组件进行了重新封装；

小程序运行在 JS Core 内，没有 DOM 树和 window 对象，小程序中无法使用 window 对象和 document 对象。

3. 请谈谈 WXSS 和 CSS 的异同?

都是用来描述页面的样式;

WXSS 具有 CSS 大部分的特性，但是也做了一些扩充和修改;

WXSS 新增了尺寸单位，WXSS 在底层支持新的尺寸单位 rpx;

WXSS 仅支持部分 CSS 选择器;

WXSS 提供全局样式与局部样式;

WXSS 不支持 window 和 dom 文档流。

4. 怎么封装微信小程序的数据请求?

方案一:

将所有的接口放在统一的 js 文件中并导出;

在 app.js 中创建封装请求数据的方法;

在子页面中调用封装的请求数据。

方案二:

在根目录下创建 utils 目录及 api.js 文件和 apiConfig.js 文件;

在 appConfig.js 封装基础的请求方法，设置请求体，带上 token 和异常处理等;

在 api.js 中引入 apiConfig.js 封装好的请求方法，根据页面数据请求的 urls，设置对应的方法并导出;

在具体页面导入。

5. 小程序页面间有哪些传递数据的方法?

在 app.js 中使用全局变量实现数据传递;

给元素添加 data-*属性来传递值，然后通过 e.currentTarget.dataset 或 onload 的 param 参数获取。注：data-名称不能有大写字母、不可以存放对象;

通过设置 id 的方法标识来传值，通过 e.currentTarget.id 获取设置的 id 的值，然后通过设置全局对象的方式来传递数值;

页面跳转或重定向时，在 navigator 中使用 url 带参数传递数据;

使用组件模板 template 传递参数;

使用缓存传递参数;

使用数据库传递参数。

6. 请谈谈小程序的双向绑定和 vue 的异同?

两者大体相同，但小程序直接使用 this.data 属性是不可以同步到视图的，必须调用 this.setData()方法。

双向绑定：vue 默认支持双向绑定；微信小程序需要借助 data 来实现。

取值：vue 中，通过 this.xxx 取值；小程序中，通过 this.data.xxx 取值。

定义方法：小程序定义方法在 page.js 中直接定义即可，vue 的方法通过写在 method 中进行定义。

取变量：小程序通过 wx:for = "{{ lists }}"；Vue 是 v-for = "item in lists";

调用 data 模型（赋值）：

小程序：this.data.item 需要调用 this.setData({item:1}) 进行赋值

vue: this.item 调用 this.item = 1 赋值

小程序的双向绑定原则上来说并不是真正的双向绑定。如果在小程序 .js 文件中改变了某个变量的值，那么页面上的值并不会跟着改变；如果想要页面上的值也跟着改变的话，需要通过 this.setData 来操作。而 Vue 默认就是双向绑定，只改变了某个变量的值，页面上也会跟着改变。

7. 请谈谈小程序的生命周期函数?

全局生命周期 app.js：

onLaunch() 小程序初始化，只会调用一次，可获取当前页面路径中的参数；

onShow() 页面显示或切入前台时触发，一般用来发送数据请求；

onHide() 页面隐藏或切入后台时触发；

onError() 页面发生错误时触发；

onPageNotFound() 小程序要打开的页面不存在时触发，可以在此函数进行重定向操作。

小程序页面的生命周期：

onLoad() 页面加载时触发，只会调用一次，可获取当前页面路径中的参数；

onShow() 页面显示或切入前台时触发，一般用来发送数据请求；

onReady() 页面初次渲染完成时触发，只会调用一次，代表页面已可和视图层进行交互；

onHide() 页面隐藏或切入后台时触发，如底部 tab 切换到其他页面或小程序切入后台等；

onUnload() 页面卸载时触发，如 redirectTo 或 navigateBack 到其他页面时；

onPullDownRefresh() 下拉刷新的钩子函数，用户下拉刷新时会自动走到这个函数中；

onReachBottom() 上拉触底的钩子函数；

onShareAppMessage() 用户点击右上角分享。

小程序组件中的生命周期：

lifetimes：组件生命周期

created() 在组件实例刚刚被创建时执行；

attached() 在组件实例进入页面节点树时执行；

ready() 在组件在视图层布局完成后执行；

moved() 在组件实例被移动到节点树另一个位置时执行；

detached() 在组件实例被从页面节点树移除时执行；

error() 每当组件方法抛出错误时执行。

pageLifetimes：组件所在页面的生命周期

show() 页面被展示时执行；

hide() 页面被隐藏时执行；

resize() 页面尺寸变化时执行。

8. 简述微信小程序原理?

小程序本质就是一个单页面应用，所有的页面渲染和事件处理，都在一个页面内进行，但又可以通过微信客户端调用原生的各种接口；

它的架构，是数据驱动的架构模式，它的 UI 和数据是分离的，所有的页面更新，都需要通过对数据的更改来实现；

它从技术上讲和现有的前端开发差不多，采用 JavaScript、WXML、WXSS 三种技术进行开发；

功能可分为 webview 和 appService 两个部分：webview 用来展现 UI，appService 用来处理业务逻辑、数据及接口调用，两个部分在两个进程中运行，通过系统层 JSBridge 实现通信，实现 UI 的渲染、事件的处理等。

9. 请谈谈原生开发小程序、wepy、mpvue 的对比?

	微信小程序	mpVue	wepy
语法规范	小程序开发规范	VueJs 开发规范	类 Vue 开发规范
标签集合	小程序标签	html 标签 + 小程序标签	小程序标签
样式规范	wxss	sass, less, postcss	sass, less, styus
组件化	无组件化机制	VueJs 组件规范	自定义组件规范
多端复用	不可复用	支持转换为 H5	支持转换为 H5
自动构建	本身无自动构建	webpack 构建	框架内置自动构建构建
上手成本	全新学习	熟悉 VueJs 即可	VueJs 和 wepy
集中数据管理	不支持	使用 Vuex 实现	不支持

个人认为，如果是新项目，且没有旧的 h5 项目迁移，则考虑用小程序原生开发，好处是相比于第三方框架，坑少。

如果有老的 h5 项目且是 vue 开发 或者 也有 h5 项目也需要小程序开发，则比较适合 wepy 或者 mpvue 来做迁移或者开发。

10. 哪些方法来提高微信小程序的应用速度?

提高页面的加载速度

用户行为预测 # 什么叫行为预测

减少默认 data 的大小

组件化方案 # 使用

11. 怎么解决微信小程序的异步请求问题?

小程序支持 ES6 语法，使用 Promise 来解决异步请求

```
function asyncFn1(){
    return new Promise(function (resolve, reject) {
        //...
    })
}
// 调用
asyncFn1()
```

.then(asyncFn2)

.then(asyncFn3);

12. 小程序关联微信公众号如何确定用户的唯一性?

使用 wx.getUserInfo 方法 withCredentials 为 true 时，可获取 encryptedData，里面有 union_id，后端需要进行对称解密。

13. 微信小程序如何实现下拉刷新?

在 json 配置中开启 enablePullDownRefresh 下拉刷新的动作，在.js 文件中通过 onPullDownRefresh 函数来实现相关的操作。

14. 微信小程序使用 webview 直接加载要注意哪些事项?

必须要在小程序后台使用管理员添加业务域名；

h5 页面跳转至小程序的版本必须是 1.3.1 以上；

微信分享只可以是小程序的主名称，如要自定义分享内容，需小程序版本在 1.7.1 以上

h5 的支付不可以是微信公众号的 appid，必须是小程序的 appid，而且用户的 openid 也必须是用户和小程序的。

15. webview 中的页面怎么跳转回小程序?

wx.miniProgram.navigateTo({

url:’ pages/login/login’ +’ $params’

```
})

// 跳转到小程序导航页面

wx.miniProgram.switchTab({

    url:’ /pages/index/index’

})
```

16. bindtap 和 catchtap 的区别?

相同点：首先他们都是作为点击事件函数，就是点击时触发。在这个作用上他们是一样的，可以不做区分。

不同点：他们的不同点主要是 bindtap 是不会阻止冒泡事件的，catchtap 是阻止事件冒泡的。

17. 简述五个路由的区别?

wx.navigateTo() 保留当前页面，跳转到应用内的某个页面。但是不能跳到 tabbar 页面；

wx.redirectTo() 关闭当前页面，跳转到应用内的某个页面。但是不允许跳转到 tabbar 页面；

wx.switchTab() 跳转到 tabBar 页面，并关闭其他所有非 tabBar 页面；

wx.navigateBack() 关闭当前页面，返回上一页面或多级页面。可通过 getCurrentPages() 获取当前的页面栈，决定需要返回几层；

wx.reLaunch() 关闭所有页面，打开到应用内的某个页面。

可以按照在 wxml、js 和跳转失败进行区分：

在 wxml 页面中：可分为 “跳转新页面”、“在当前页打开”、“切换到首页 Tab”；

在 js 页面中：分为 “应用内的页面” 和 “tabBar 页面” ；

如果上述跳转遇到跳转失败或者无效的问题，请访问：wx.navigateTo/wx.redirectTo 无效

18. 微信小程序与 H5 的区别?

运行环境不同：小程序在微信运行，h5 在浏览器运行；

开发成本不同：h5 需要兼容不同的浏览器；

获取系统权限不同：系统级权限可以和小程序无缝衔接；

应用在生产环境的运行流畅度：h5 需不断对项目优化来提高用户体验；

19. 小程序如何更新页面中的值?

可以通过 this.setData 来进行改变。

20. 如何实现登录数据的持久化?

可以通过 wx.setStorageSync('键名'，对应的值）来进行数据持久化存储。

21. 微信小程序和 app 有什么不同之处?

微信小程序属于轻量级的 app 但是限制在微信中，开发周期短，功能较少，占用空间少；

app 就相反 需要占用额外内存 开发周期长

22. 微信小程序如何进行双向绑定?

通过 bind-tap 点击事件 向 app.js 定义的方法中获取回执;

设置 data 的值 实现双向绑定。

23. 如何自定义 tabbar?

取消当前的 tabbar

插入自定义 tabbar 组件

app.json 调用组件

页面显示 tabbar 组件

24. 小程序怎样使用自定义组件?

新建自定义组件目录，生成目录结构;

写好组件内容;

在要使用的目标页面的 json 文件中配置下 usingComponents，引入组件;

以标签的形式在页面中使用该组件即可;

传递数据和 vue 一样，通过自定义属性，然后在组件里通过 properties 接收就可以使用了。

25. 小程序本地存储（数据缓存）有哪些常用 api?

数据的存储，将数据存储在本地缓存中指定的 key 中

wx.setStorage(Object object) 异步

wx.setStorageSync(string key, any data) 同步

数据的获取，从本地缓存中异步获取指定 key 的内容

wx.getStorage(Object object) 异步

wx.getStorageSync(string key) 同步

存储信息的获取，异步获取当前 storage 的相关信息

wx.getStorageInfo(Object object) 异步

wx.getStorageInfoSync() 同步

数据的删除，从本地缓存中移除指定 key

wx.removeStorage(Object object) 异步

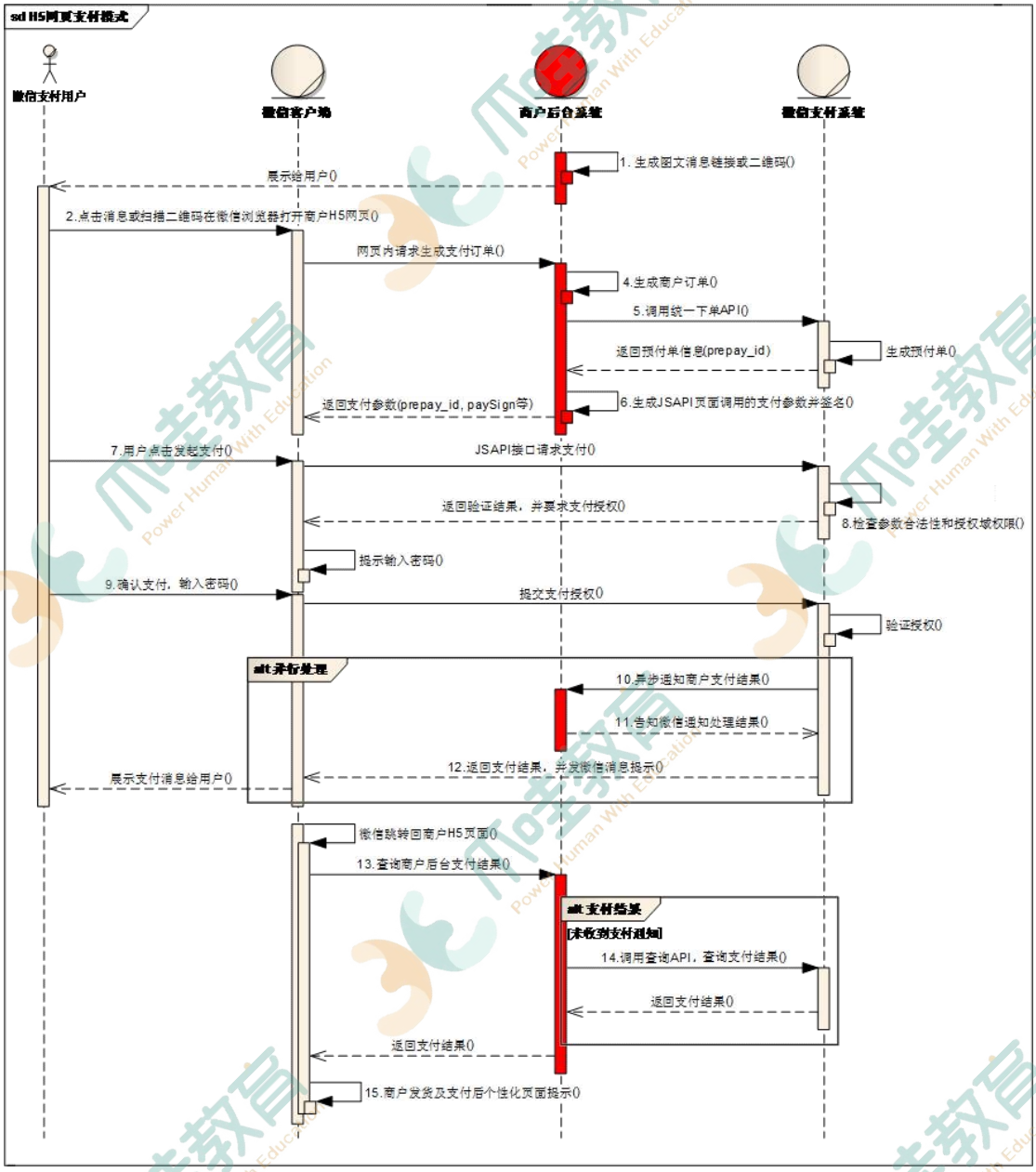
wx.removeStorageSync(string key) 同步

数据的清空，清理本地数据缓存

wx.clearStorage(Object object) 异步

wx.clearStorageSync() 同步

26. 微信支付的流程简单说一下?



27. 如何自定义 tabBar?

现在小程序官方提供了 自定义 tabBar 的能力，根据官方提供的 demo 很容易就可以实现自定义 tabBar。

但是要实现 tab 选中态，要在当前页面下，通过 getTabBar 接口获取组件实例，并调用 setData 更新选中态。

下载官方提供的 demo，然后合并到自己的代码中。

配置信息：在 app.json 中的 tabBar 项指定 custom 字段，同时其余 tabBar 相关配置也补充完整。

所有 tab 页的 json 里需声明 usingComponents 项，也可以在 app.json 全局开启。

添加 tabBar 代码相关文件。

编写 tabBar 代码：用自定义组件的方式编写即可，该自定义组件完全接管 tabBar 的渲染。另外，自定义组件新增 getTabBar 接口，可获取当前页面下的自定义 tabBar 组件实例。

28. 说一下小程序页面之间的传值?

URL 传值

这种方式最常用，比如通过 wx.navigateTo({}) 直接通过跳转页面的 URL 进行传值：

```
wx.navigateTo({
  url: '../detail/detail?cid=' + cid + '&access_token=' + access_token;
})
```

然后在另一个页面通过 options 进行接收：

```
onLoad: function (options) {
  console.log('cid =' + options.cid);
```

```
  console.log('access_token =' + options.access_token);
}
```

这种传值方式只适合值比较少的时候使用。

本地缓存

传值比较多的时候，建议写本地缓存进行传值。

小程序 API 提供了本地缓存数据的 API，默认可以缓存 10M 的数据：

```
wx.setStorageSync('checkin', checkin);
```

在需要的页面直接调用 wx.getStorageSync 即可获取到存储的本地缓存数据。

全局 app

我们可以利用 app.js 和 app.wxss 中的代码都是全局生效的这一特性，在不同页面之间进行传值。

```
App({
  // 全局变量
  globalData: {
    host: 'https://xxx/xcx',
    version: 2,
    versionFeature: '更新说明'
  }
})
```

在使用的页面中通过引入 app.js 来使用定义的全局变量。

const app = getApp();

let app_host = app.globalData.host;

29. 本地图片资源无法通过 wxss 获取?

本地资源图片无法通过 WXSS 获取，可以使用网络图片，或者 base64，或者使用标签来解决。

30. wx.navigateTo 无法打开页面是为什么？如何解决?

一个应用同时只能打开 5 个页面，当已经打开了 5 个页面之后，wx.navigateTo 不能正常打开新页面。请避免多层级的交互方式，或者使用 wx.redirectTo。

31. tabBar 设置不显示?

tabBar 设置不显示有如下几个原因：

tabBar 的数量少于 2 项或超过 5 项都不会显示；

tabBar 写法错误导致不显示；

tabBar 没有写 pagePath 字段（程序启动后显示的第一个页面）

32. 小程序调用后台接口遇到哪些问题?

数据的大小有限制，超过范围会直接导致整个小程序崩溃，除非重启小程序；

小程序不可以直接渲染文章内容页这类型的 html 文本内容，若需显示要借助插件，但插件渲染会导致页面加载变慢，所以最好在后台对文章内容的 html 进行过滤，后台直接处理批量替换 p 标签 div 标签为 view 标签，然后其它的标签让插件来做，减轻前端处理的时间。

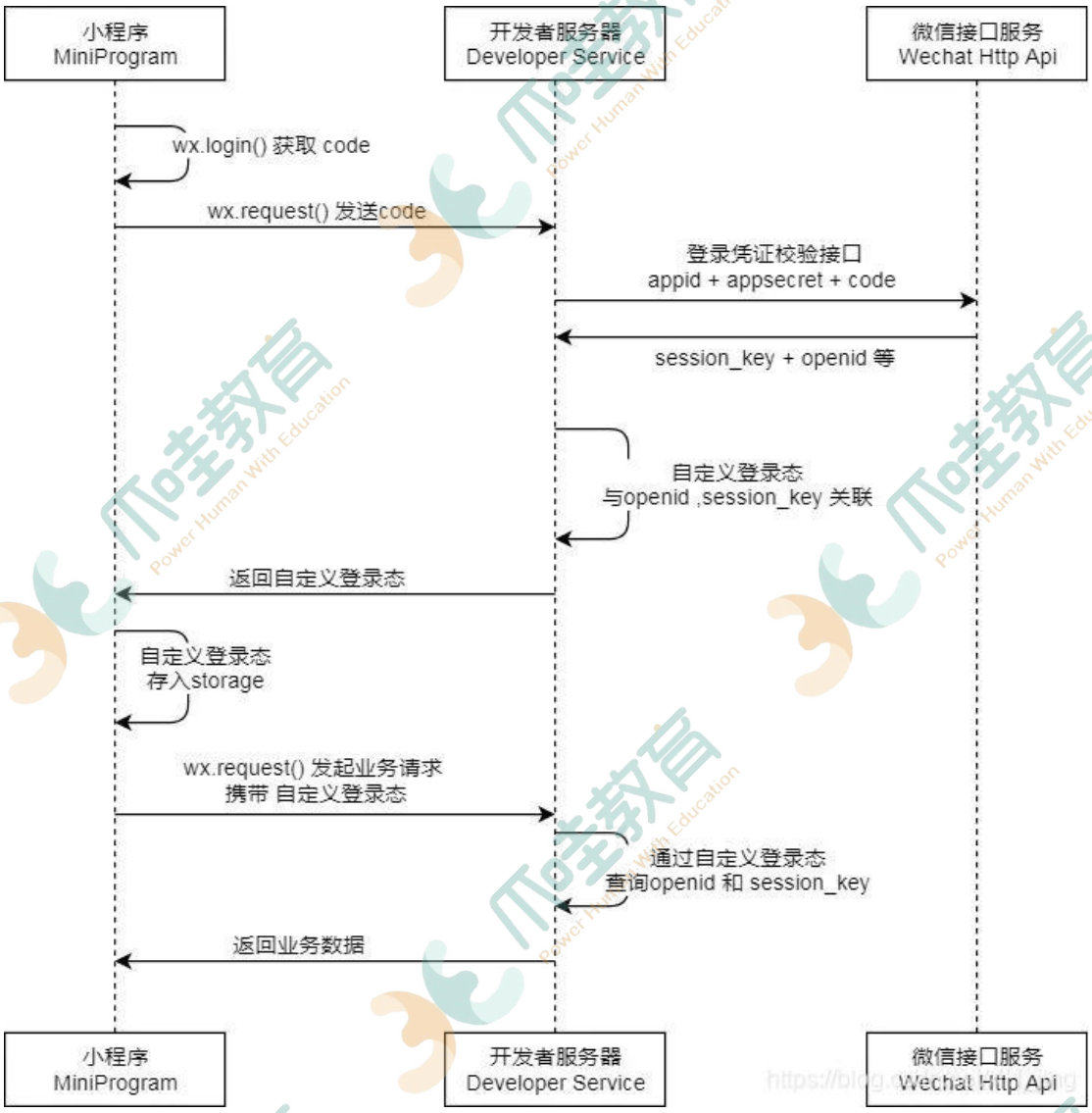
33. 小程序的登录流程?

调用 wx.login() 获取 code

调用 wx.request() 发送 code 到我们自己的服务器（我们自己的服务器会返回一个登录状态的标识，比如 token）

将登录状态的标识 token 进行存储，以便下次使用

请求需要登录状态的接口时，带上这个 token。



34. 请谈谈微信小程序作用?

小程序是一种不需要下载安装即可使用的应用，它实现了应用“触手可及”的梦想，用户扫一扫或者搜一下即可打开应用。也体现了“用完即走”的理念，用户不用关心是否安装太多应用的问题。应用将无处不在，随时可用，但又无需安装卸载。对于开发者而言，小程序开发门槛相对较低，难度不及 APP，能够满足简单的基础应用，适合生活服务类线下商铺以及非刚需低频应用的转换。

33. 分析微信小程序的优劣势?

优点:

可进行各种功能开发，比传统网页实现的功能更全面;

小程序支持模糊搜索，在符合关键字的搜索结果中小程序就能展现。小程序可以与公众号关联，公众号能给小程序带来流量，同时小程序也能给公众号带来粉丝;

平台流量大：依托于微信平台，用户基数大，使用频繁，使得小程序在社交分享方面有天然的优势;

线上广告推广，广告成本低，降低企业成本;

加载速度：小程序加载速度要快，小程序的基本元素是不需要加载的;

安全性：小程序因为代码都上传到了微信服务器上，所以安全性是比较高的。

缺点:

微信小程序对包大小有限制，单个包最大不能超过 2M，这样导致无法开发一些大型的小程序。但是微信小程序正在努力解决这方面的问题，不到扩大分包的数量;

小程序的技术框架还不稳定，开发方法时常有修改，导致短时间内经常要升级维护;

不能跳转外链网址，所以间接影响了小程序的开放性;

不能直接分享到朋友圈，少了一个重要的推广方式;

需要像 APP 一样审核上架，这点比 HTML5 即做即发布要麻烦些，不过审核也比较快。

六、Vue 面试题（附答案）

1. keep-alive 组件有什么作用?

keep-alive 是 vue 的内置组件，而这个组件的作用就是能够缓存不活动的组件。一般情况下，组件进行切换的时候，默认是会进行销毁的，如果我们有需求，在某个组件切换后不进行销毁，而是保存之前的状态，那么就可以利用 keep-alive 来实现。

在 keep-alive 上有两个属性，可以对字符串或正则表达式进行匹配，匹配到的组件会被缓存。

include 值为字符串或者正则表达式匹配的组件 name 会被缓存。（缓存匹配到的组件）

exclude 值为字符串或正则表达式匹配的组件 name 不会被缓存。（排除匹配到的组件）

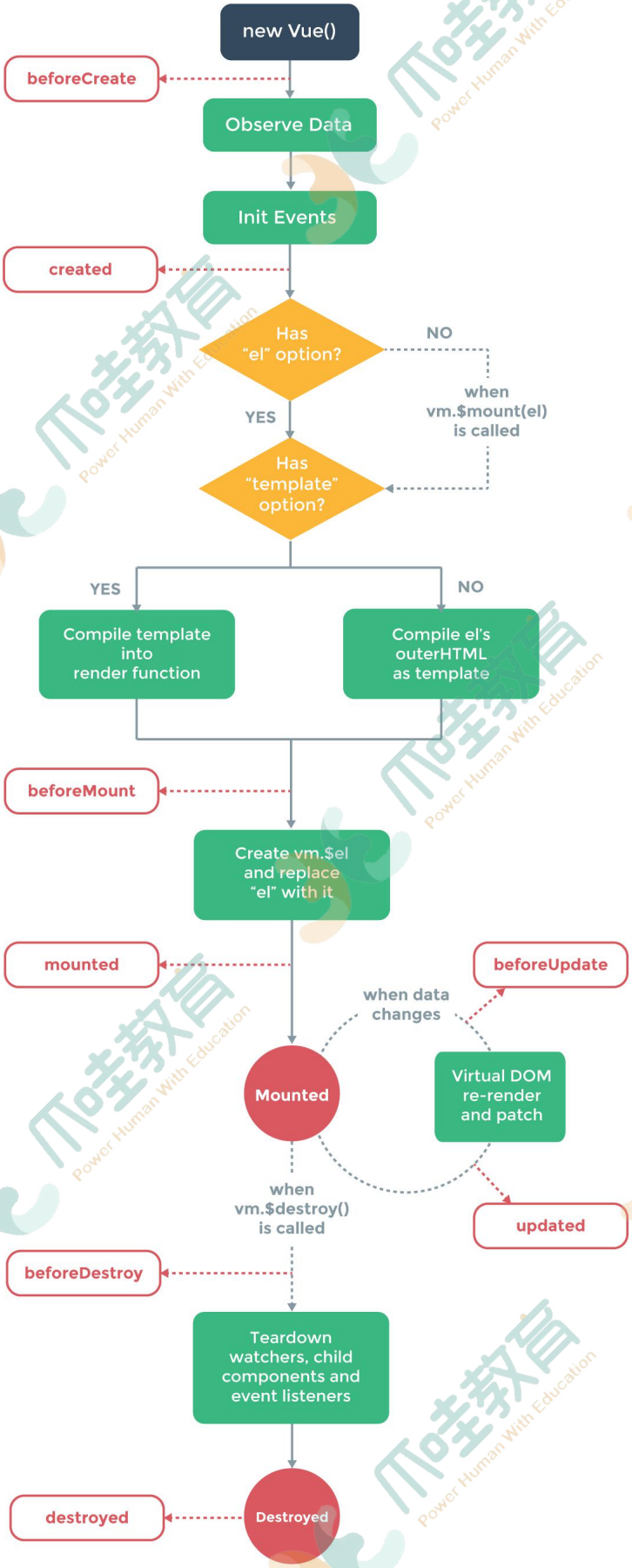
其拥有两个独立的生命周期钩子函数 actived 和 deactived，使用 keep-alive 包裹的组件在切换时不会被销毁，而是缓存到内存中并执行 deactived 钩子函数，命中缓存渲染后会执行 actived 钩子函数。

2. 说下 vue 生命周期钩子函数?

每个 vue 实例在被创建时都要经过一系列的初始化过程。

所有的生命周期钩子自动绑定 this 上下文到实例中，因此可以在函数中访问数据，对属性和方法进行运算。这意味着不能使用箭头函数来定义一个生命周期方法【这是因为箭头函数绑定了父上下文，因此 this 与你期待的 Vue 实例不同】。

vue 的生命周期图：


new Vue()
beforeCreate
Observe Data
Init Events
created
Has "el" option?
NO
when vm.$mount(el) is called
YES
Has "template" option?
YES
NO
Compile template into render function
Compile el's outerHTML as template
beforeMount
Create vm.$el and replace "el" with it
mounted
beforeUpdate
when data changes
Mounted
Virtual DOM re-render and patch
when vm.$destroy() is called
updated
beforeDestroy
Teardown watchers, child components and event listeners
destroyed
Destroyed

阶段一：Vue 实例创建阶段

beforeCreate

Vue 实例在内存中刚被创建，this 变量还不能使用，数据对象（data）和方法（methods）未初始化，watcher 中的事件都不能获得到；

created

实例已经在内存中创建好，数据和方法已经初始化完成，但是模板还未编译，页面还是没有内容，还不能对 dom 节点进行操作（此时访问 this.$el 和 this.$refs.xxx 都是 undefined）

beforeMounte

找到对应的 template 模板，编译成 render 函数，转换成虚拟 dom，此时模板已经编译完成，数据未挂载到页面，也就是说在这个阶段你可以看到标签间的双花括号，数据还未渲染到页面中；

render : h=>h(App)

在 beforeMounte 之后和 mounted 之前，还有渲染 render 函数，它的作用是把模板渲染成虚拟 dom。

mounted

模板编译好了，虚拟 dom 渲染成真正的 dom 标签，数据渲染到页面，此时 Vue 实例已经创建完毕，如果没有其他操作的话，Vue 实例会静静的躺在内存中，一动不动。

一般会在 mounted 中来渲染从后端获取的数据。(页面初始化时，如果有操作 dom 的事件一般也会放在 mounted 钩子函数中。当然，也可以放在 create 中，前提需使用 this.$nextTick(function(){})，在回调函数中操作 dom。)

阶段二：Vue 实例运行阶段

beforeUpdate

数据依赖改变或者用 $forceUpdata 强制刷新时，对象 data 中的数据已经更改（虚拟 dom 已经重新渲染)，但是 页面中的值还是原来，未改变，因为此时还未开始渲染 dom；

update

此时 data 中的数据和页面更新完毕，页面已经被重新渲染。

在实际开发中，一般会用监听器 watch 来代替上边 2 个方法，因为 watch 会知道是哪一个数据变化。

阶段三：Vue 实例销毁阶段

beforeDestroy

实例销毁前使用，在此刻实例还是可用的。

destroyed

Vue 实例被销毁，观察者、子组件、事件监听被清除（页面数据不会消失，只不过是响应式无效了）。

3.vue 组件中 data 必须是一个函数

如果 data 是一个对象，当复用组件时，因为 data 都会指向同一个引用类型地址，其中一个组件的 data 一旦发生修改，则其他重用的组件中的 data 也会被一并修改。

如果 data 是一个返回对象的函数，因为每次重用组件时返回的都是一个新对象，引用地址不同，便不会出现如上问题。

4.Vue 中 v-if 和 v-show 有什么区别?

v-if 在进行切换时，会直接对标签进行创建或销毁，不显示的标签不会加载在 DOM 树中。

v-show 在进行切换时，会对标签的 display 属性进行切换，通过 display 不显示来隐藏元素。

一般来说，v-if 的性能开销会比 v-show 大，切换频繁的标签更适合使用 v-show。

5.Vue 中 computed 和 watch 有什么区别?

计算属性 computed：

支持缓存，只有依赖数据发生变化时，才会重新进行计算函数；

计算属性内不支持异步操作；

计算属性的函数中都有一个 get(默认具有，获取计算属性)和 set(手动添加，设置计算属性)方法；

计算属性是自动监听依赖值的变化，从而动态返回内容。

侦听属性 watch：

不支持缓存，只要数据发生变化，就会执行侦听函数；

侦听属性内支持异步操作；

侦听属性的值可以是一个对象，接收 handler 回调，deep，immediate 三个属性；

监听是一个过程，在监听的值变化时，可以触发一个回调，并做一些其他事情。

6.Vue-router 路由有哪些模式?

一般有两种模式：

hash 模式：后面的 hash 值的变化，浏览器既不会向服务器发出请求，浏览器也不会刷新，每次 hash 值的变化会触发 hashchange 事件。

history 模式：利用了 HTML5 中新增的 pushState() 和 replaceState() 方法。这两个方法应用于浏览器的历史记录栈，在当前已有的 back、forward、go 的基础之上，它们提供了对历史记录进行修改的功能。只是当它们执行修改时，虽然改变了当前的 URL，但浏览器不会立即向后端发送请求。

7.Vue-cli 项目中 assets 和 static 文件夹有什么区别?

两者都是用于存放项目中所使用的静态资源文件的文件夹。其区别在于：

** assets 中的文件在运行 npm run build 的时候会打包**，简单来说就是会被压缩体积，代码格式化之类的。打包之后也会放到 static 中。static 中的文件则不会被打包。

七、React 篇

1、React 中的 key 是什么，有什么作用

key 是 react 在渲染一系列相同类型的兄弟元素时，给每个元素指定一个稳定、可预测、兄弟间唯一的值，来帮助 React 识别哪些元素改变了，比如添加和删除，这样做可以避免在某些场景下的错误渲染并且提升 React 的渲染性能

key 的作用是用在使用 diff 算法对比 react 更新前后两棵树的比较时使用的，使得树的转换效率得以提高，组件实例基于它们的 key 来决定是否更新以及复用

2、组件通信

组件间通信也就是两个或多个组件之间相互传递消息。从关系上来划分：

父级向子级传递信息

子级向父级传递信息

同级之间传递信息

跨组件传递信息

React 是一种单向数据流的设计，也就是说信息只能从父级向子级一层一层向下传递

1）父级向子级进行通信：这种信息传递比较简单，父组件在调用子组件时，只需要把想传递的数据加在子组件的属性上，然后在子组件内部通过 props 属性来接收就可以了。之后，每次 props 属性发生变化，子组件都会重新进行渲染的

2）子组件向父组件传递信息：React 是单向数据流，没有办法从子组件直接传递信息到父组件，但是可以在父组件上定义好回调后，把回调函数通过 props 属性传递给子组件，利用回调向父组件传递信息

3）同级组件之间的信息传递：通过状态提升，将信息共同保存在父组件中，在通过回调函数更改信息后在用 props 传递给子组件

4）跨层级组件通信：利用高级 API-context 上下文的方式，这里直接转到我的另外一篇博客里面有详细介绍 React 高级 API-Context

5）Redux、React-redux 等状态管理库结合异步请求处理 redux-thunk、redux-saga

3、React 合成事件机制

React 中有自己的事件系统模式，通常被称为 React 合成事件。之所以采用这种自己定义的合成事件，一方面是为了抹平事件在不同平台体现出来的差异性，这使得 React 开发者不需要自己再去关注浏览器事件兼容性问题；另一方面是为了统一管理事件，提高性能，这主要体现在 React 内部实现事件委托，并且记录当前事件发生的状态上

事件委托，也就是我们通常提到的事件代理机制，这种机制不会把事件处理函数直接绑定在真实的节点上，而是把所有的事件绑定到结构的最外层，使用一个统一的事件监听和处理函数。当组件挂载或卸载时，只是在这个统一的事件监听器上插入或删除一些对象；当事件放生时，首先被这个统一的事件监听器处理，然后在映射表里找到真正的事件处理函数并调用。这样做简化了事件处理和回收机制。效率也有很大的提升

记录当前事件发生的状态，即记录事件执行的上下文，这便于 React 来处理不同事件的优先级，达到谁优先级高先处理谁的目的，这里也就实现了 React 的增量渲染思想，可以预防掉帧，同时达到页面更加顺滑的目的，提升用户体验

4、细读 setState

setState(updater, [callback])

其将对组件的 state 的更改排入队列，并通知 React 需要使用更新后的 state 重新渲此组件及其子组件。这是用于更新用户界面以及响应事件处理器和处理服务器数据的主要方式

React 会延迟调用 setState，然后通过一次传递更新多个组件，不会保证 state 的变更立即生效，如果要立即拿到 this.state 的值，可以在其回调函数 callback 或者 componentDidUpdate 里取到最新的值

参数 updater 有两种形式：

第一种是函数形式

this.setState((state, props) => stateChange)

第二种是对象形式，会进行批量更新，如果要使用前一次的 state 值，请使用函数形式

this.setState({value:2})

setState 中的异步与同步：

异步是指批量更新，达到性能优化的目的，在合成事件和生命周期中都是异步的，不能立马拿到 this.state 的值

同步是指立马能拿到最新的 this.state 值，在定时器 setTimeout 和原生事件中是同步的，另外在其第二个参数回调函数 callback 中和 componentDidUpdate 更新生命周期函数中也能得到最新的 this.state 值

5、函数组件与 class 组件如何选择

1、hook 之前的函数组件是什么样子

无状态、无副作用，只能做单纯的展示组件

2、class 组件有什么弊端，为什么要引入 hook

- 在组件之间复用状态逻辑很难
- 复杂组件变得难以理解
- 难以理解的 class

3、引入了 hook 之后的函数组件发生了哪些变化

函数组件可以存储和改变状态值(useState、useReducer)，可以执行副作用(useEffect、useLayoutEffect)，还可以复用状态逻辑(自定义 hook)

4、函数组件与 class 组件如何选择

出现以上缺点的情况下都适合使用函数组件，函数组件与 class 组件无缝对接

6、React 性能优化方案

1、减少不必要渲染，如用 shouldComponentUpdate、PureComponent、React.memo 实现

2、数据缓存

- useMemo 缓存参数、useCallback 缓存函数
- 函数、对象尽量不要使用内联形式(如 context 的 value object、refs function)
- Route 中的内联函数渲染时候使用 render 或者 children，不要使用 component，当你用 component 的时候，Router 会用指定的组件和 React.createElement 创建一个新的[React elment]。这意味着当你提供的是一个内联函数的时候，每次创建 render 都会创

建一个新的组件。这会导致不再更新已经现有组件，而是直接卸载然后再去挂载一个新的组件。因此，当用到内联函数的内联渲染时，请使用render 或者 children

3、不要滥用功能项，如 context、props 等

4、懒加载，对于长页表分页加载

5、减少 http 请求

7、React-router 路由模式?

hash 模式（HashRouter）：通过监听 hashchange 事件，在回调里拿到 window.location.hash 的值。 hash 就是指 url 尾巴后的 # 号以及后面的字符。

hash 模式原理：

使用 window.location.hash 属性及窗口的 onhashchange 事件，可以实现监听浏览器地址 hash 值变化，执行相应的 js 切换网页。 hash 指的是地址中#号以及后面的字符，也称为散列值。

history 模式（BrowserRouter）： 利用 history API 实现 url 地址改变，网页内容改变。

history 模式原理：

window.history 属性指向 History 对象，它表示当前窗口的浏览历史。 History 对象保存了当前窗口访问过的所有页面网址。

八．Node.js 篇

1．在每个 tick 的过程中，如何判断是否有事件需要处理呢?

1.每个事件循环中有一个或者多个观察者，而判断是否有事件需要处理的过程就是向这些观察者询问是否有要处理的事件。

2.在 Node 中，事件主要来源于网络请求、文件的 I/O 等，这些事件对应的观察者有文件 I/O 观察者，网络 I/O 的观察者。

3.事件循环是一个典型的生产者/消费者模型。异步 I/O，网络请求等则是事件的生产者，源源不断为 Node 提供不同类型的事件，这些事件被传递到对应的观察者那里，事件循环则从观察者那里取出事件并处理。

4.在 windows 下，这个循环基于 IOCP 创建，在*nix 下则基于多线程创建

2.请描述一下整个异步 I/O 的流程

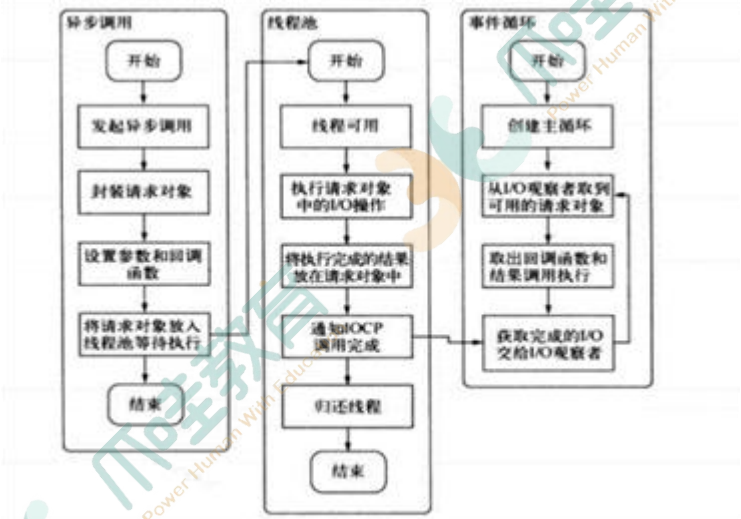


图3-13　整个异步I/O的流程

3.V8 的内存限制是多少，为什么 V8 这样设计

64 位系统下是 1.4GB， 32 位系统下是 0.7GB。因为 1.5GB 的垃圾回收堆内存，V8 需要花费 50 毫秒以上，做一次非增量式的垃圾回收甚至要 1 秒以上。这是垃圾回收中引起

Javascript 线程暂停执行的事件，在这样的花销下，应用的性能和影响力都会直线下降。

4.Node. js 中的事件循环是什么样的?

事件循环其实就是一个事件队列，先加入先执行，执行完一次队列，再次循环遍历看有没有新事件加入队列。

执行中的事件叫 IO 事件，setImmediate 在当前队列中立即执行，setTimout/setInterval 把执行定时到下一个队列， process. nextTick 在当前队列执行完，下次遍历前执行。所以总体顺序是： IO 事件→ setImmediate→ setTimeout/setInterval→ process. nextTick。

5.Node. js 的优缺点是什么?

优点如下：

（1） Node. js 是基于事件驱动和无阻塞的，非常适合处理并发请求，因此构建在 Node. js 的代理服务器相比其他技术实现的服务器要好一点。

（2）与 Node. js 代理服务器交互的客户端代码由 JavaScript 语言编写，客户端与服务端都采用一种语言编写。

缺点如下：

（1） Node .js 是一个相对新的开源项目，不太稳定，变化速度快。

（2）不适合 CPU 密集型应用，如果有长时间运行的计算（比如大循环），将会导致 CPU 时间片不能释放，使得后续 I/O 无法发起。

6.npm 的作用是什么?

npm 是同 Node .js 一起安装的包管理工具，能解决 Node. js 代码部署上的很多问题。常见的使用场景有以下几种。

（1）允许用户从 npm 服务器下载别人编写的第三方包到本地。

（2）允许用户从 npm 服务器下载并安装别人编写的命令行程序到本地。

（3）允许用户将自己编写的包或命令行程序上传到 npm 服务器供别人使用。

7.有哪些常用 Stream 流？分别什么时候使用？

Readable 流为可读流，在作为输入数据源时使用；Writable 流为可写流，在作为输出源时使用；Duplex 流为可读写流，它作为输出源被写入，同时又作为输入源被后面的流读出。Transform 流和 Duplex 流一样，都是双向流，区别是 Transfrom 流只需要实现一个函数 _transfrom(chunk, encoding, callback)；而 Duplex 流需要分别实现_read(size)函数和_write(chunk, encoding, callback ）函数。

8.Nginx 和 Apache 有什么区别？

Nginx 是轻量级的，同样的 Web 服务在 nginx 中会占用更少的内存和资源。Nginx 抗并发，处理请求的方式是异步非阻塞的，负载能力比 Apache 高很多，而 Apache 则是阻塞型的。

在高并发下 Nginx 能保持低资源、低消耗、高性能，并且处理静态文件比 Apache 好。

Nginx 的设计高度模块化，编写模块相对简单，配置简洁。作为负载均衡服务器，支持 7 层负载均衡，是一个反向代理服务器。

社区活跃，各种高性能模块出品迅速。Apache 的 rewrite 比 Nginx 强大，模块丰富。Apache 发展得更为成熟，Bug 很少,更加稳定。

Apache 对 PHP 的支持比较简单， Nginx 需要配合其他后端使用。Apache 处理动态请求有优势，拥有丰富的特性、成熟的技术和开发社区。

9、说说线程与进程的区别。

（1）一个程序至少有一个进程，一个进程至少有一个线程

（2）线程的划分尺度小于进程，使得多线程程序的并发性高。

（3）进程在执行过程中拥有独立的内存单元，而多个线程共享内存，极大地提高了程序的运行效率。

（4）线程在执行过程中与进程有区别。每个独立的线程都有程序运行的入口、顺序执行序列和程序的出口。但是线程不能够独立执行，必须依存在应用程序中，由应用程序提供多个线程执行控制。

（5）从逻辑角度来看，多线程的意义在于一个应用程序中，有多个执行部分可以同时执行。但操作系统并没有将多个线程看作多个独立的应用来实现进程的调度、管理和资源分配。这是进程和线程的主要区别。

10.node 中的异步和同步怎么理解

node 是单线程的，异步是通过一次次的循环事件队列来实现的．同步则是说阻塞式的 IO,这在高并发环境会是一个很大的性能问题，所以同步一般只在基础框架的启动时使用，用来加载配置文件，初始化程序什么的

11.有哪些方法可以进行异步流程的控制?

1）多层嵌套回调

2) 为每一个回调写单独的函数，函数里边再回调

3) 用第三方框架比方 async, q, promise 等