评论(1)

>>

# 移动端H5最佳实践

### REM单位布局适配方案

在页面开发中, rem (Root EM)是一种相对单位,用于在网页设计和开发中定义元素的尺寸和间距。它基于HTML根元素(通常是 <html> )的字体大小,而不是直接依赖于父元素的字体大小,比如,如果根元素的字体大小是16px,那么1rem就等于16px。

这使得REM在响应式设计中非常有用,因为所有的尺寸都是基于一个统一的参考点(即根元素的字体大小),只需修改根元素的字体大小,整个页面的比例会自动调整。

### 常规方案

通常移动端设计稿的尺寸为 750px

在代码中直接使用 rem 进行样式的开发,需要进行以下2个步骤

#### 实现过程

#### 1. 初始化根元素字体大小

通过JavaScript读取屏幕宽度,然后根据宽度计算出对应的尺寸并设置根元素的 font-size 。

以下是需要调整至1rem=100px的初始化代码。

- 1 // 确定设计稿宽度和基准值,即屏幕物理像素和设计稿一致的情况下,根元素font-size设置成多少如果1822
- 2 const designWidth = 750;
- 3 const remBase = 50;
- 4 // 根据当前屏幕宽度得到最终的根元素font-size
- 5 const rootFontSize = window.innerWidth / designWidth \* remBase;
- 6 // 更新根元素的font-size
- 7 document.documentElement.style.fontSize = rootFontSize + "px";

设置完成后,浏览器中的font-size在iphone6s(宽度375px)中计算结果为50px。





根元素的font-size不宜过小,这是由于部分旧机型和浏览器不支持小于12px的字体,小于12px仍按12px去计算,从而导致实际尺寸偏大的问题。

#### 屏幕物理像素和设计稿一致



刘滨 2024年11月19日 (编辑过)

这一标准是 iphone6s 时期约定的,那时候大部分设备 dpr 都是 2,沿用至今,直接按 2 倍计算即可。

iphone6s

物理像素: 750px 实际像素: 375px 设计稿尺寸: 750px

因此 1:100 的换算比例下计算出 来的根元素 font-size 是 50px

#### 2. 设计稿单位换算

基于第1步的设置,我们可以知道在写尺寸单位的时候,需要拿设计稿尺寸除以 100 再加上 rem。

```
1 // 设计稿尺寸
2 .logo {
3  padding: 24px;
4 }
5
6 // 开发实际编写尺寸
7 .logo {
8  padding: 0.24rem;
9 }
```

### 不足之处

- 1. 写样式需要自行换算数值,相对直接写 px 而言不是那么直观
- 2. 第三方组件(vant )、迁移代码(从其它项目复制过来的组件)中的样式文件不方便直接改造,导致出现 rem + px 混合使用的情况
- 3. 所有样式写死了 rem 单位,如果将来有更好的样式方案(如目前同样流行的vw),或者样式需要迁移到小程序(小程序单位需要使用 rpx ,不支持 rem ),由于所有的尺寸数值必须改变,不好改造

### 改良措施

### 1. 通过编译将px转换为rem

基于常规方案的不足之处,我们希望通过编译解决数值换算的问题,在开发过程中使用和设计稿一致的数值+px 来写代码,在编译产物中全部转换成 rem 尺寸。

借助postcss-pxtorem可以做到,这是 postcss 的插件,和打包工具 (vite / webpack )无关。

基于常规方案第1步的设置,编译后代码1rem=设计稿100px,计算出来的font-size是50px,进行如下配置

```
1 // main.js
2 // 确定设计稿宽度和基准值,即屏幕物理像素和设计稿一致的情况下,根元素font-size设置成多少
3 const designWidth = 750; // 设计稿尺寸基准
4 const remBase = 50; // 根元素font-size, 由于实际宽度缩小一倍,算出来是25
5 // 根据当前屏幕宽度得到最终的根元素font-size
6 const rootFontSize = window.innerWidth / designWidth * remBase;
7 // 更新根元素的font-size
8 document.documentElement.style.fontSize = rootFontSize + "px";
```

```
1 // vite: vite.config.js属性链路为css -> postcss
2 // webpack: 到postcss-loader下配置
3 export default {
4 // ...
5 postcss: {
6 plugins: [
7 pxtorem({
```

配置完成后,尺寸单位即可按照设计稿的数值写px进行开发,此外,所有经过入口导入和编译的第三方包也会统一编译为 rem 单位。

### 2. 注意统一第三方组件和页面的尺寸数值和单位

注意下面这张表,如果设计稿尺寸基准是750px,插件的rootValue设为50,经过转换后 屏幕中的实际像素是代码的2倍

方案	设计稿宽 度	根元素 font-size	375px转换后数 值/实际像素	100px转换后数值 / 实际像素	1px转换后数值 / 实际像素
2倍屏 设计稿	750рх	25рх	7.5rem / 187.5px	2rem / 50px	0.02rem / 0.5px
实际像 素	375px	100px	3.75rem / 375px	1rem / 100px	0.01rem / 1px

这就带来一个问题,有些第三方组件例如 vant@2 是 px 单位写尺寸的,如果根元素按 50px ,它的所有元素渲染出来的实际尺寸都会变为原来的一半,达不到预期效果。如 果按 100px 去转换,则没有这个问题。

因此,面对这种情况,需要事先考虑好,设计稿的尺寸基准应该选择750还是375。

最终解决方案:可以将vant的样式按另一个基准去转换

## 输入框交互体验

# 手机号和验证码输入框拉起数字键盘

```
1 <input type="tel" pattern="[0-9]*" max-length="11" />
2 <input type="number" pattern="[0-9]*" max-length="6" />
```





type="number"

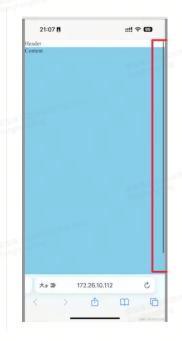
### 短信验证码自动填充

需要手机已授权开启了自动识别短信验证码功能,一般都是默认开启的 必须设置maxlength,否则可能会出现验证码重复填充的问题

1 <input type="number" maxlength="6" autocomplete="one-time-code" />

# 动态视口适配

在IOS safari浏览器中,导航栏和工具栏是悬浮在页面上方的,这导致当我们想设置一个元素的高度为100vh时(通常用于做局部滚动),这个元素会比页面的视口(innerWidth)更高,导致出现滚动条。



对此,w3c推出了新的单位:dvh(动态视口高度),实际上就是动态的window.innerHeight

最佳的解决方案是使用100dvh代替100vh,但是它对旧机型的兼容性不好



但可以借助JS辅助实现类似的效果,代码如下

```
document.body.style.setProperty("--app-height", "100dvh");
    if (!CSS.supports("height", "100dvh")) {
      // 如果不支持,就定义 --app-height 为视口高度,即 window.innerHeight
3
      document.body.style.setProperty("--app-height",
4
    window.innerHeight + "px");
      // 当屏幕缩放时,改变内容高度。因为 resize 事件触发很频繁,所以使用节流
    减少性能损耗
      let timeout;
6
7
      function onDynamicHeightResize() {
        clearTimeout(timeout);
8
9
        timeout = setTimeout(() => {
          clearTimeout(timeout);
10
11
          document.body.style.setProperty(
            "--app-height",
12
13
            window.innerHeight + "px"
          );
14
15
        }, 500);
16
17
      window.addEventListener("resize", onDynamicHeightResize);
18
19
```

```
<html lang="zh-CN" style="font-size: 130px;"> (scroll)

> <head> (head> (head> )

> <hoody style="--app-height: 963px;"> == $0

> <div> (div> (head> )

> <script> (script> )

<script src="//res.wx.qq.com/open/js/jweixin-1.4.0.js" (/script> )

<script type="module" src="/src/main.js?t=173649906685 (/script> )

<script src="//service.100bt.com/wx/jsapi5ignature.jsc e=15&noncestr=0667022_52Ftopup%252Faqsy%252F%253Ffrom% allback=callback_7129275688618224"> </script> (!---> )
```

低版本浏览器

高版本浏览器

#### 在CSS中,只需要这么写即可

```
1 .conainer {
2    height: var(--app-height);
3 }
```

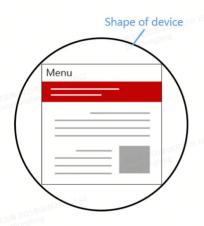
# iPhoneX底部安全区适配

### 1. html头部设置viewport-fit=cover

viewport-fit的3个值:

- contain: 可视窗口完全包含网页内容(左图)
- cover: 网页内容完全覆盖可视窗口(右图) 🗸

• auto: 默认值,此值不影响初始布局视图端口,并且整个web页面都是可查看的。





### 2. 在需要预留安全区的元素设置CSS

注意顺序不能错

可以添加paading-bottom,或者放一个空白的 <div class="safe-area"> 元素

```
1 .safe-area {
2    @supports (bottom: constant(safe-area-inset-bottom)) or
    (bottom: env(safe-area-inset-bottom)) {
3         height: constant(safe-area-inset-bottom); /*兼容 IOS<11.2*/
4         height: env(safe-area-inset-bottom); /*兼容 IOS>11.2*/
5     }
6 }
```

#### 最终效果







无微信导航栏

有微信导航栏

浏览器

## 微信动态导航栏标题设置

微信访问H5时,无法直接通过 document.title 动态更改导航栏的标题,通常借助 iframe来更新,封装方法如下

```
1
    const setIosWechatTitle = (title) => {
       document.title = title;
2
       const iframe = document.createElement("iframe");
3
      iframe.src = "/";
       iframe.style.display = "none";
6
       iframe.onload = function () {
7
         setTimeout(function () {
8
          iframe.remove();
10
        }, 0);
11
      document.body.appendChild(iframe);
12
13
    };
14
    export function usePageTitle(title) {
15
      if (isIOS) {
16
         setIosWechatTitle(title);
17
       } else {
18
         document.title = title;
19
20
21
22
```

# 禁用横屏提示



#### 1. 根组件放置代码

#### 2. 引用CSS

```
.wap-orient-tips {
1
2
       position: fixed;
3
       top: 0;
       left: 0;
      width: 100%;
 5
       height: 100%;
6
 7
       z-index: 9999;
       background: rgba(0, 0, 0, 0.9);
8
9
       flex-direction: column;
       align-items: center;
10
       justify-content: center;
11
       display: none;
12
       color: #fff;
13
14
     .wap-orient-tips .inner {
15
16
       display: flex;
17
       flex-direction: column;
       align-items: center;
18
19
       justify-content: center;
20
     .wap-orient-tips .inner::after {
21
      content: "为了更好的体验,请将手机/平板竖过来";
22
23
       font-size: 10px;
24
     .wap-orient-tips img {
25
26
      width: 60px;
      height: 60px;
27
      margin-bottom: 4px;
28
29
       object-fit: contain;
       animation: rotateAni 1.5s ease infinite alternate;
30
31
     .wap-orient-tips span {
32
    font-size: 10px;
33
34
    @media screen and (min-aspect-ratio: 4 / 3) and (max-width:
35
    992px) {
       .wap-orient-tips {
36
         display: flex;
37
38
39
40
    @keyframes rotateAni {
41
         transform: rotate(-90deg);
42
43
      }
       30% {
44
45
        transform: rotate(-90deg);
       }
46
       70% {
47
        transform: rotate(0deg);
48
49
       }
       100% {
50
       transform: rotate(0deg);
52
53
54
```

# 阻止页面缩放

页面缩放有2种方式触发:

- 双指捏合缩放
- 双击屏幕放大

以前我们可以通过meta阻止缩放:

```
代码块

1 <meta
2 name="viewport"
3 content="width=device-width, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0, viewport-fit=cover"
4 />
```

但是在IOS 10以上的版本,这行代码已经不管用了,需要通过JS对上述2点手势操作进行 处理

```
代码块
1 // 处理IOS 10以上可双指放大和可双击放大的问题
    window.onload = function () {
      // 阻止双击放大
      let lastTouchEnd = 0;
      document.addEventListener("touchstart", function (event) {
        if (event.touches.length > 1) {
          event.preventDefault();
        }
8
9
      });
      document.addEventListener(
10
11
        "touchend",
        function (event) {
12
13
          const now = new Date().getTime();
          if (now - lastTouchEnd <= 300) {</pre>
14
            event.preventDefault();
15
          }
16
          lastTouchEnd = now;
17
18
        },
19
        false
      );
20
      // 阻止双指放大
21
      document.addEventListener("gesturestart", function (event) {
22
23
        event.preventDefault();
24
      });
25
    };
26
```

## 点击反馈

如果页面对动效有一定要求,通常需要加入一些点击反馈让操作更加跟手,可以通过伪类实现

#### 你的米姆偷偷帮别人浇菜,你决定

暴揍一顿

劝米姆回家

```
代码块
1 // 点击时出现阴影遮罩
    .click--highlight {
      position: relative;
      overflow: hidden;
5
6
      &:after {
        content: "";
7
        position: absolute;
8
9
        top: 0;
10
        left: 0;
11
        width: 100%;
        height: 100%;
12
        background-color: rgba(0, 0, 0, 0.3);
13
        z-index: 100;
14
        opacity: 0;
15
16
        transition: opacity 0.2s ease-in-out;
      }
17
18
      &:active:after {
19
        opacity: 1;
20
21
22
    }
23
    // 点击时元素变小
24
     .click--smaller {
25
    transition: transform 0.1s linear;
26
27
28
      &:active {
29
        transform: scale(0.9);
30
      }
31
   }
```

在PC模拟器下我们可以看到:active伪类的效果,但是在IOS下仍然未生效,原因是IOS会 先判定touchstart事件(长按的情况),从而让active伪类没有立即生效,因此需要通过 JS阻止touchstart的判定,增加以下代码即可解决

```
代码块
1
2
   // 让css的:active伪类在移动端下也生效
3
   document.addEventListener(
     "touchstart",
4
5
     function () {
       return false;
6
7
     },
     false
8
   );
```

### 自动抓取图片和字体资源进行预加载

#### 原理:

- 1. 提前将需要渲染的资源通过JS请求一遍(期间显示loading动画),在浏览器二次加载时会命中缓存从而免去二次网络请求
- 2. window.onload时通过抓取head中StyleSheet的内容,可将所有匹配到后缀的资源收集起来进行预加载

好处:避免用户在使用过程中因资源未加载完成导致的画面抖动

执行以下方法即可进行资源的预加载(非阻塞,加载失败仍然按成功处理)

#### 源码如下:

```
代码块
    // 预加载图片
    const preloadImage = (url: string): Promise<void> => {
      return new Promise((resolve, reject) => {
        const img = new Image();
        img.onload = () => resolve();
        img.onerror = reject;
        img.src = url;
7
8
      });
9
    };
10
    // 预加载字体
11
    const preloadFont = (url: string): Promise<void> => {
12
13
      return new Promise((resolve, reject) => {
         const font = new FontFace("preload-font", `url(${url})`);
14
        font
15
           .load()
16
           .then(() => resolve())
17
           .catch(reject);
18
      });
19
20
    };
21
    // 从CSS中提取资源URL
22
23
    const extractUrlsFromCSS = (cssText: string): { images: string[];
     fonts: string[] } => {
      const images: string[] = [];
24
      const fonts: string[] = [];
25
26
       // 匹配url()中的图片
      const imageRegex = /url\(['"]?([^'"()]+\.
28
     (?:png|jpg|jpeg|gif|webp|svg))['"]?\)/gi;
29
      let match;
      while ((match = imageRegex.exec(cssText)) !== null) {
30
31
         images.push(match[1]);
32
33
      // 匹配@font-face中的字体
34
```

```
const fontRegex = /@font-face\s*{[^}]*url\(['"]?([^'"()]+\.
35
     (?:woff2?|ttf|eot|otf))['"]?\)[^}]*}/gi;
      while ((match = fontRegex.exec(cssText)) !== null) {
36
         fonts.push(match[1]);
37
38
      }
39
40
      return { images, fonts };
41
    };
42
    // 主预加载函数
43
    export const preloadAssets = async (): Promise<boolean> => {
44
     30// 获取所有样式表
45
      const styleSheets = Array.from(document.styleSheets);
46
       console.log({ styleSheets });
47
      let cssText = "";
48
49
       // 收集所有CSS文本
50
      styleSheets.forEach(sheet => {
51
52
        try {
           const rules = Array.from(sheet.cssRules);
53
54
           rules.forEach(rule => {
             cssText += rule.cssText;
55
          });
56
        } catch (e) {
57
           console.warn("无法读取样式表:", e);
58
59
      });
60
61
       // 提取资源URL
62
       const { images, fonts } = extractUrlsFromCSS(cssText);
63
64
65
      console.log(cssText);
66
67
       console.log({
68
        cssText,
69
        images,
70
        fonts
71
      });
72
73
      const taskLength = images.length + fonts.length;
74
75
      let progress = 0;
76
      const preloadTask = async (task: () => Promise<void>) => {
77
         try {
78
           await task();
79
          console.log("资源加载中", `${progress + 1}/${taskLength}`);
          progress++;
80
        } catch (err) {
81
           console.error("预加载资源失败:", err);
82
83
84
         return true;
85
      };
86
      // 预加载所有资源
87
      const preloadPromises = [
88
         ...images.map(url => preloadTask(() => preloadImage(url))),
89
         ...fonts.map(url => preloadTask(() => preloadFont(url)))
90
91
      ];
92
      await Promise.all(preloadPromises);
93
      console.log("资源加载完成");
94
95
```

```
96 68778 1627

188 20258 068778 1622

288 20258 068778 1622

288 20258 068778 1622

288 20258 068778 1622

288 20258 068778 1622

288 20258 068778 1622
```

由于它只抓取CSS中的资源,因此需要注意以下2点

- 1. 需要在CSS中引入资源,html中引入的获取不到
- 2. 对于vite打包的项目,在vue SFC中写的CSS,为了按需加载做了拆包,项目初始化时可能获取不到,因此建议写成单入口的形式,统一通过 /src/styles/index.less 来引入

# 分享信息优化

	lianghonBre		lianghongie	
环境	微	(信	QQ	浏览器(分享到 微信/QQ)
	公众号网页	非公众号网页		
优化方式	对接微信jssdk	无,只支持获取 title	IOS:  1. 标题和描述动态 获取当前页面的 <title>, &lt;meta name="descrip tion&gt;&lt;/td&gt;&lt;td&gt;获取当前页面的&lt;br&gt;TDK,建议按照&lt;br&gt;Open Graph(开&lt;br&gt;放图谱)协议 配&lt;br&gt;置&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;家族後 2025年06月27日 16:22&lt;br&gt;litanghonglens&lt;/td&gt;&lt;td&gt;架光镜 2025年06月27日 16:22&lt;br&gt;Wanghongten8&lt;/td&gt;&lt;td&gt;與恶性 2025年06月27日 16:22&lt;br&gt;Manghongleng&lt;/td&gt;&lt;td&gt;2. 封面图获取当前&lt;br&gt;页面DOM中的第&lt;br&gt;一个img元素src&lt;/td&gt;&lt;td&gt;&lt;meta name="og:title" &gt; &lt;meta name="og:descr&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;京法院 2025年06月27日 16:22&lt;br&gt;Itanghangtens&lt;/td&gt;&lt;td&gt;東洲縣 2025年06月27日 16:22&lt;br&gt;東洲縣 dongfens&lt;/td&gt;&lt;td&gt;類法據 2025年06月27日 16:22&lt;br&gt;lianghongtens&lt;/td&gt;&lt;td&gt;• 分享时远程获取&lt;br&gt;线上url的TDK,&lt;br&gt;不支持动态设置&lt;br&gt;• 线上url获取不到&lt;br&gt;时,用标题:&lt;/td&gt;&lt;td&gt;iption"&gt;&lt;br&gt;&lt;meta&lt;br&gt;name="og:imag&lt;br&gt;e"&gt;&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;是基础 2025年06月27日 16:22&lt;br&gt;Battle 2025年08月27日 16:22&lt;br&gt;Banghongleng&lt;/td&gt;&lt;td&gt;家没有 2025年06月27日 16:22&lt;br&gt;Nanghongfen8&lt;/td&gt;&lt;td&gt;與法据 2025年06月27日 16-22&lt;br&gt;lianghongleng&lt;/td&gt;&lt;td&gt;"分享"、封面&lt;br&gt;图: "当前截&lt;br&gt;图"和描述: url&lt;br&gt;兜底&lt;/td&gt;&lt;td&gt;說別題 2025年06月27日 16:22&lt;br&gt; langhongfeng&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;是否支持运&lt;br&gt;行时动态更&lt;br&gt;改&lt;/td&gt;&lt;td&gt;支持&lt;br&gt;&lt;sup&gt;BRX W 2025-BDD B 27EI 16:22&lt;/sup&gt;&lt;br&gt;&lt;sub&gt;Hanghongleng&lt;/sub&gt;&lt;/td&gt;&lt;td&gt;不支持&lt;/td&gt;&lt;td&gt;IOS支持(但可能有&lt;br&gt;缓存)&lt;br&gt;Android不支持&lt;/td&gt;&lt;td&gt;支持&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;分享效果&lt;/td&gt;&lt;td&gt;一起測測种葉 MBTI拿大奖!&lt;br&gt;我的种田 MBTI 竟然是…&lt;br&gt;快来测测你的吧!还有机&lt;br&gt;会赢取农场承包权、上…&lt;/td&gt;&lt;td&gt;酶,在吗?随陷我玩米婷吧!&lt;br&gt;在线征集同频的搭子~&lt;/td&gt;&lt;td&gt;NOT A STATE OF THE STATE OF THE&lt;/td&gt;&lt;td&gt;一起測測种菜MBTI拿大奖!&lt;br&gt;我的种田MBTI與然是&lt;br&gt;快来週期你的吧! 还有机&lt;br&gt;会赢取农场承包权、上&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;念故版 2025年06月27日 16:22&lt;br&gt;Janghongtens&lt;/td&gt;&lt;td&gt;宗光镜 2025年06月27日 16:22&lt;br&gt;Wanghongtens&lt;/td&gt;&lt;td&gt;完活版 2025年06月27日 16:22&lt;br&gt;lianghongteng&lt;/td&gt;&lt;td&gt;IOS Androi&lt;br&gt;d线上地&lt;br&gt;址不存&lt;br&gt;在&lt;/td&gt;&lt;td&gt;一起測測仲菜MBTI拿大奖1&lt;br&gt;分享文集持衛は&lt;br&gt;の分部分享  教的种菜MBTI居然是派对永&lt;br&gt;动机,来看看你的吧1&lt;br&gt;(米別年間) ― (本田&lt;br&gt;好友 上記憶的音大楽1&lt;br&gt;日第初末年季号添)&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;於基準 2025年06月27日 16-22&lt;br&gt;Banghongjeng&lt;/td&gt;&lt;td&gt;東洲博 2025年06月27日 16:22&lt;br&gt;Hanghongfen8&lt;/td&gt;&lt;td&gt;與為權 2025年06月27日 16-22&lt;br&gt;lianghongleng&lt;/td&gt;&lt;td&gt;《洛克王国: 世界》 《洛克王国: 世界》 《洛克王国: 世界》 《洛克王国: 世界》 是所稱 廣方工管理的是由現象的洛 克王国*中续作。玩家将在与 《乱涂! 彩世界》官方网站 一一放肆换色,近回**中冲游 神观委员会一级警报! 外星 人入侵! 均址正在画临庆色 危机打工恶感和不靠遗神—&lt;/td&gt;&lt;td&gt;の外数分享&lt;br&gt;最高に関する5年06月27日 16:22&lt;br&gt;においません。&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;聚准律 2025年06月27日 16:22&lt;br&gt;Nanghongteng&lt;/td&gt;&lt;td&gt;&lt;/td&gt;&lt;td&gt;安卓线上地址存在且&lt;br&gt;设置了TDK&lt;/td&gt;&lt;td&gt;&lt;/td&gt;&lt;/tr&gt;&lt;/tbody&gt;&lt;/table&gt;</title>	

#### 结论:

- 1. 针对IOS端QQ分享的场景,建议页面中设置第一个img标签为封面图,样式设置为隐藏,避免客户端抓取到无关的图片当封面
- 2. 针对Android端QQ分享的场景,需要增加<meta name="og:image">(第3点有提到)
- 3. 我们以前常常忽略了从浏览器分享到App的场景,建议优化按照Open Graph协议增加 <meta name="og:title"><meta name="og:description"><meta name="og:image">等信息