**{ 性能优化 }**

**一、图片性能优化**

|  |
| --- |
| 1. ## 图片使用场景 2. \* jpg -- 大部分不需要透明图片的业务场景 3. \* png -- 大部分需要透明图片的业务场景 4. \* webp -- 安卓全部 5. \* gif -- 动态图业务场景 6. \* svg矢量图 -- 图片样式相对简单的业务场景 7. \* 精灵图 -- 小图颜色比较丰富 8. \* 字体图标 -- 页面图标，颜色单一 9. \* base64 -- 用于12kb以下图片 |

**懒加载：**

|  |
| --- |
| 1. var viewHeight = document.documentElement.clientHeight; 2. function lazyLoad() { 3. var imgNodes = document.querySelectorAll('.lazyLoading'); 4. imgNodes.forEach(function (item) { 5. const { top } = item.getBoundingClientRect(); 6. if ( top < viewHeight ) { 7. const newSrc = item.dataset.src; 8. const image = new Image(); 9. image.src = newSrc; 10. image.onload = function () { 11. item.src = newSrc; 12. }; 13. item.className = ''; 14. item.removeAttribute('data-src'); 15. } 16. }) 17. } 18. lazyLoad(); 20. document.onscroll = lazyLoad; |

**预加载：**

|  |
| --- |
| 1. //建立数组，遍历，新建图片来进行缓存 2. var arr = ['./imgs/1.jpg', './imgs/2.jpg', './imgs/3.jpg']; 3. arr.forEach((item) => { 4. const image = new Image(); 5. image.src = img; 6. }) |

**二、重排重绘**

1. ## 渲染过程
2. \* 浏览器渲染页面的整个过程：浏览器会从上到下解析文档。
3. 1. 遇见 HTML 标记，调用HTML解析器解析为对应的 token （一个token就是一个标签文本的序列化）并构建 DOM 树（就是一块内存，保存着tokens，建立它们之间的关系）。
4. 2. 遇见 style/link 标记 调用css解析器 处理 CSS 标记并构建 CSSOM 树。
5. 3. 遇见 script 标记 调用 javascript解析器 处理script标记，绑定事件、修改DOM树/CSSOM树 等
6. 4. 将 DOM 与 CSSOM 合并成一个渲染树。
7. 5. 根据渲染树来布局，以计算每个节点的几何信息。 (重排)
8. 6. 将各个节点绘制到屏幕上。（重绘）

## 触发重绘的属性

\* color \* background \* outline-color

\* border-style \* background-image \* outline

\* border-radius \* background-position \* outline-style

\* visibility \* background-repeat \* outline-width

\* text-decoration \* background-size \* box-shadow

## 触发重排(回流)的属性

盒子模型相关属性 定位属性及浮动 改变文字结构

\* width \* top \* text-align

\* height \* bottom \* overflow-y

\* padding \* left \* font-weight

\* margin \* right \* overflow

\* display \* position \* font-family

\* border-width \* float \* line-height

\* border \* clear \* vertival-align

## 优化手段

\* 元素位置移动变换时尽量使用CSS3的transform来代替对top left等的操作

\* 将多次改变样式属性的操作合并成一次class操作（修改DOM的className）

\* 将DOM离线后再修改

\* 由于display属性为none的元素不在渲染树中，对隐藏的元素操作不会引发其他元素的重排。

\* 如果要对一个元素进行复杂的操作时，可以先隐藏它，操作完成后再显示。这样只触发了2次重排。

\* 不要把某些DOM节点的属性值放在一个循环里当成循环的变量

\* 当你请求向浏览器请求一些 style信息的时候，就会让浏览器刷新队列，比如：

\* offsetTop, offsetLeft, offsetWidth, offsetHeight

\* scrollTop/Left/Width/Height

\* clientTop/Left/Width/Height

\* width,height

\* 当你请求上面的一些属性的时候，浏览器为了给你最精确的值，需要刷新队列，

\* 因为队列中可能会有影响到这些值的操作。即使你获取元素的布局和样式信息跟最近发生或改变的布局信息无关，浏览器都会强行刷新渲染队列。

\* 动画实现过程中，启用GPU硬件加速

\* transform: translateZ(0)

\* 利用文档碎片

\* 不要使用table布局

## window.requestAnimationFrame()

\* 方法告诉浏览器您希望执行动画并请求浏览器在下一次重绘之前调用指定的函数来更新动画

1. var divNode = document.querySelector('div');
2. let dis = 1;
3. function move(callback) {
4. dis++;
5. if(dis >= 500) return;
6. divNode.style.transform = `translateX(${ dis }px)`;
7. requestAnimationFrame(move);
8. }
9. requestAnimationFrame(move);

**三、webStorage**

•本地存储：（离线存储，5M左右）

localStorage：(本地存储--持久存储)

localStorage.setItem("key","value");

localStorage.getItem("key");

localStorage.removeItem("key");

localStorage.clear();

sessionStorage:(临时存储—关闭网站自动删除)

•跨域通信（通过获取改变localStorage/ sessionStorage）

window.addEventListener('storage', function (e) {

console.log(e)

})

新能优化网站：

<https://csspod.com/frontend-performance-best-practices/> 是、