### 性能优化

- ◆ 是一个综合性问题,没有标准答案,但要求尽量全面
- ◆ 某些细节问题可能会单独提问:手写防抖、节流
- ◆ 只关注核心点,针对面试

性能优化原则

- ◆ 多使用内存、缓存或其他方法
- ◆ 减少 CPU 计算量,减少网络加载耗时
- ◆ (适用于所有编程的性能优化 —— 空间换时间)

从何入手

- ◆ 让加载更快
- ◆ 让渲染更快

### 让加载更快

- ◆ 减少资源体积:压缩代码
- ◆ 减少访问次数:合并代码, SSR 服务器端渲染, 缓存
- ◆ 使用更快的网络:CDN

#### 让渲染更快 - 1

- ◆ CSS 放在 head , JS 放在 body 最下面
- ◆ 尽早开始执行 JS ,用 DOMContentLoaded 触发
- ◆ 懒加载(图片懒加载,上滑加载更多)

## 让渲染更快 - 2

- ◆ 对 DOM 查询进行缓存
- ◆ 频繁 DOM 操作,合并到一起插入 DOM 结构
- ◆ 节流 throttle 防抖 debounce

### 资源合并

```
1 <script src="a.js"></script>
2 <script src="b.js"></script>
3 <script src="c.js"></script>
```

1 <script src="abc.js"></script>

## 缓存

```
module.exports = {
    mode: 'production',
    entry: path.join(__dirname, 'src', 'index'),
    output: {
        filename: 'bundle.[contenthash].js',
        path: path.join(__dirname, 'dist')
    },
```

```
distJs bundle.58cf28598ed2e217d4b3.jsJs bundle.jsindex.html
```

#### 缓存

- ◆ 静态资源加 hash 后缀,根据文件内容计算 hash
- ◆ 文件内容不变,则 hash 不变,则 url 不变
- ◆ url 和文件不变,则会自动触发 http 缓存机制,返回 304

### **CDN**

href="https://cdn.bootcss.com/bootstrap/4.0.0-alpha.6/css/bootstrap.css" rel="sty ot src="https://cdn.bootcss.com/zepto/1.0rc1/zepto.min.js"></script> 新闻 hao123 地图 视频 贴配 学术 [k ] Elements Console Sources Network Performance ● 🛇 😽 Q 🛛 Preserve log 🗆 Disable cache | Online 🔻 | Hide data URLs All XHR JS CSS Img Type Initiator jquery-1.10.2\_1c4228b8.js ss0.bdstatic.com/5aV1bjqh... sbase\_a65ca873.js (index) min\_super\_fdb28b91.js ss0.bdstatic.com/5aV1bjqh... script (index)
Parser all\_async\_search\_76d7e43.js ss1.bdstatic.com/5eN1bjq8... script (index):80 り1回 百 every\_cookie\_mac\_82990d... ss1.bdstatic.com/5eN1bjq8... script jauery-1.10.2... Script nu\_instant\_search\_068a951.js ss1.bdstatic.com/5eN1bjq8... swfobject\_0178953.js script jquery-1.10.2... Script

#### **SSR**

- ◆ 服务器端渲染:将网页和数据一起加载,一起渲染
- ◆ 非 SSR (前后端分离):先加载网页,再加载数据,再渲染数据
- ◆ 早先的 JSP ASP PHP , 现在的 vue React SSR

### 懒加载

```
1  <img id="img1" src="preview.png" data-realsrc="abc.png"/>
2  <script type="text/javascript">
3      var img1 = document.getElementById('img1')
4      img1.src = img1.getAttribute('data-realsrc')
5  </script>
```

### 缓存 DOM 查询

```
// 不缓存 DOM 查询结果
for (let = 0; i < document.getElementsByTagName('p').length; i++) {
    // 每次循环, 都会计算 length , 频繁进行 DOM 查询
}

// 缓存 DOM 查询结果
const pList = document.getElementsByTagName('p')
const length = pList.length
for (let i = 0; i < length; i++) {\}
    // 缓存 length , 只进行一次 DOM 查询
}</pre>
```

## 多个 DOM 操作一起插入到 DOM 结构

```
const listNode = document.getElementById('list')

// 创建一个文档片段,此时还没有插入到 DOM 树中
const frag → document.createDocumentFragment()

// 执行插入
for(let x = 0; x < 10; x++) {
    const li = document.createElement("li")
    li.innerHTML = "List item " + x
    frag.appendChild(li)
}

// 都完成之后,再插入到 DOM 树中
listNode.appendChild(frag)
```

# 尽早开始 JS 执行