1. 综述：
   1. 页面渲染原理：
      1. ①解析HTML，生成DOM树；解析CSS，生成CSSOM树（并行）
      2. ②将DOM树和CSSOM树结合，生成渲染树(Render Tree)
      3. ③Layout：根据生成的渲染树，将所有的节点在平面上合成
      4. ④Painting：根据渲染树以及回流得到的几何信息，得到节点的绝对像素
      5. ⑤Display:将像素发送给GPU，展示在页面上。（GPU会将多个合成层合并为同一个层并展示在页面中。css3硬件加速的原理是新建合成层）
      6. 回流与重绘：
         1. 回流（reflow）：渲染树因为元素的改变而需要重新构建，重新触发上述③。回流结束后一定会触发重绘
         2. 重绘（repaint）：渲染树的一些元素需要更新属性，重新触发上述④，且这些属性只会影响外观，风格，不影响布局
         3. 浏览器一般有队列机制，将多次修改存储起来一次性回流，但获取布局信息时会强制清空队列进行回流。
   2. DOM文档加载步骤：
      1. 解析HTML结构、加载并解析外部脚本、DOM树构建完成并执行脚本、（DOMContentLoaded）、
      2. 加载图片及样式表等外部文件、页面加载完毕、（load）
   3. Attribute与Property：
      1. attribute的本质为DOM的属性节点；property的本质为DOM对象的一个属性。attribute中一般有一个相对应的property属性，它们共享数据，一般情况下数据相同，布尔类型除外。checked=“checked”【attr】 ==> checked：true【prop】
   4. 响应式设计：视口宽度不同显示不同样式
      1. 移动端优先、渐进增强（移动端满足后逐渐满足pc端）
   5. SEO优化：
      1. title网站标题：<title>网站名（产品名）-网站介绍（不超过30字）</title>
      2. description网站说明：<meta name=”description” content=”...” />
      3. keywords关键字：<meta name=”keywords” content=”...” />
      4. logo的优化：logo中放一个h1标签，h1中放一个链接（返回首页）并把背景图片给链接，链接里放文字（网站名称）但不显示（textindent
      5. - 移到盒子外并overflow：hidden 或 font-size：0），给链接title属性来设置提示文字
   6. 单页面应用：single page application（SPA）
      1. 特点：将所有活动局限于一个页面；当页面中有部分数据发生了变化，不会刷新整个页面，而是局部刷新（利用ajax技术和路由）
   7. H5本地存储技术：（**value若为不为字符串，自动调用toString，所以对象一般转换为JSON**）
      1. sessionStorage：会话存储
         1. 生命周期：浏览器打开到关闭的过程。大小：5MB。保存在浏览器端
         2. 方法：[window.]sessionStorage
            1. setItem（‘key’，value） / getItem（‘key’） / removeItem（‘key’）/ clear（）
      2. localStorage：永久存储
         1. 生命周期：永久，除非人为删除。大小：5MB甚至更大。保存在浏览器端
         2. 方法：[window.]localStorage
            1. setItem（‘key’，value）/ getItem（‘key’） / removeItem（‘key’）/ clear（）
      3. cookie：用于浏览器和服务器端进行通信。
         1. 特点：最大4kb；每次发送请求都携带，占用带宽；保存在浏览器端；容易被截获；生命周期：会话cookie、人为设置cookie
   8. 进程与线程：
      1. 进程（process）：是一块内存空间， 负责为程序的运行提供必备的环境
      2. 线程（thread）：是进程内的一个独立执行单元；是程序执行的一个完整流程；是CPU的最小调度单元
         1. 应用程序必须运行在某个进程的某个线程上；一个进程中至少有一个运行的线程（主线程，在进程启动后自动创建）；
         2. 一个进程中可以同时运行多个线程，则该程序为多线程运行的；一个进程内的数据可以同其中多个线程直接共享；
         3. 多个线程之间的数据不能直接共享
      3. 线程池（thread pool）：保存多个线程对象的容器，实现线程对象的反复利用
      4. 多线程：优点：提高CPU的利用率。缺点：存在创建多线程开销；存在线程间切换开销；死锁与状态同步问题
      5. 单线程：优点：循序编程简单易懂。缺点：效率低
   9. 路由：路由是一组映射关系（key-value）
      1. 前端路由：key为路径，value为Component，用于展示页面内容。可实现单页面应用，见vue-router库
      2. 后端路由：key为路径，value为function，用于处理客户端提交的请求。
         1. 工作过程：服务器接收请求时，根据请求路径找到函数处理请求，返回数据
   10. 浏览器对于历史记录的保存方式：栈结构
       1. push模式：每次打开新的网页（路径更改也算），push到栈中，可以返回上次路径
       2. replace模式：每次打开新的网页（路径更改也算），替换栈顶元素，无法返回上次路径
   11. url的hash模式和history模式：
       1. #(hash)模式：url中#右面的字符不会通过http请求发送到服务器；若将地址通过app分享，校验严格时不合法；兼容性好
       2. history模式：url中没有# ；兼容性略差（路由路径在刷新时会被解析成普通路径，出现问题）；要由后端解决：
          1. connect-history：npm i connect-history-api-fallback
             1. express框架服务器的server.js中：

const history = require（‘npm i connect-history-api-fallback’）

app.use（history（））

1. 移动端：
   1. 像素：
      1. CSS像素：虚拟像素
      2. 物理像素：显示器上的像素
      3. CSS像素与物理像素比：是在宽度或高度一个方向上的比值
      4. 默认情况在pc端，一个css像素对应一个物理像素
      5. 智能手机的像素点远小于计算机屏幕的像素点，用一个css像素对应多个物理像素以保证显示效果合适
   2. 视口（viewport）：就是浏览器显示内容的区域
      1. 视口一般不看高度，只看宽度。html元素是视口的子元素
      2. 浏览器通过改变css像素和物理像素的比例实现缩放，也可以通过调整视口大小来调整CSS物理像素比
      3. 一般情况下，移动端的视口大小为980px（css像素），如果网页的宽度超过980，移动端的浏览器会自动对网页缩放以完整显示网页
      4. 每一个移动设备在设计时，都有一个最佳像素比，将像素比设置为最佳状态时，视口为完美视口
      5. 设置完美视口：<meta name=”viewport” content=”width=device-width，initial-scale=1.0, user-scalable=no”>
      6. 因为完美视口的大小因设备而异，px单位不再能满足需求，因此引入：
         1. vw单位（viewport width）：100vw=一个视口的宽度
            1. vw适配：设计图为750px=100vw，换算得0.133333vw=1px

设置html的font-size：0.133333vw；

设置元素大小时：设计图中大小为48px，css设置width：48rem

设置字体仍然用px