1:你做过哪些性能优化?

- html性能优化

1:语意化标签　　(利于搜索引擎收录　也提高了代码阅读性)

2:减少标签的嵌套　　(利于渲染引擎的加速渲染)

3:把script标签移到HTML文件末尾，因为JS会阻塞后面的页面的显示

4:减少iframe的使用，因为iframe会增加一条http请求，阻止页面加载，即使内容为空，加载也需要时间

5:减少注释，因为过多注释不光占用空间，如果里面有大量关键词会影响搜索引擎的搜索。上线的时候最好清空注释

6:html结构的复用性

- css性能优化

1:尽量不使用通配符(*)

2:把页面公用的样式给封装到css文件里面(减少代码量、复用性强、便于维护)

3:不要用内联样式和行类样式(比较乱　维护性比较差)

4:尽量使用最具体的类别、避免后代选择器、属于标签类别的规则永远不要包含子选择器
//因为从左到右解析关系，在CSS选择器中后代选择器非但没有帮我们加快CSS查找。反而查找很慢

5:能用css实现的样式就不要用图片。减少http请求

5:合并压缩css文件　减少http请求

- js性能优化

1:可以多使用事件委托　减少事件注册

2:多使用if语句多做判断　减少不必要的操作

3:创建标签的时候尽量通过字符串创建，不要通过createElement。因为通过createElement创建完之后还要再次操作这个dom去设置属性和文本等。而直接创建字符串只需要一步设置好。

4:设置样式的时候　尽量不要用style去设置样式　维护性 可读性差　最好添加通过class去设置样式

5:尽量少使用eval函数。(使用eval相当于在运行时再次调用解释引擎对内容进行运行，需要消耗大量时间，而且使用Eval带来的安全性问题也是不容忽视的。)

6:使用三目运算符替代条件分支可以减少解释器对条件的探测次数

7:提高代码复用性使用封装。减少代码量、维护性好。

8:合并压缩js 减少http请求

- 图片性能优化

1:合并图片生成雪碧图 减少http请求

2:图片小于2KB转换成base64 通过css去设置　减少http请求

3:图片压缩大小

4:如果不考虑兼容问题　可以通过canvas代替图片

5:配合js使用懒加载。加载图片

2:公司项目开发流程

1:产品根据需求做出产品原型 --> 后台

2:ui根据产品原型做设计稿 psd

3:前端是根据设计稿作出页面

4:集成开发 + 前后台交互

5:测试-->改bug 1:本地测试 2:灰度(bate)测试 3:线上测试

6:上线-->夜晚12点

3.电商项目负责模块有哪些？购物车商品数据存在什么地方?支付功能?

1: 自己总结一下找到电商网站有哪些页面?

2: 购物车数据一般存在后台。不能用本地存储，因为换一台浏览器的话 本地储存的数据就用不了。而且商品的价格是会改变的。如果用本地存储，价格改变了拿不到最新数据。

3:通过用ping++去实现

4:说说模块化 commonjs、AMD、CMD不同?

> AMD/CMD/CommonJs是JS模块化开发的标准，目前对应的实现是RequireJs/SeaJs/nodeJs。CommonJs主要针对服务端，AMD/CMD主要针对浏览器端，所以最容易混淆的是AMD/CMD。

AMD/CMD区别

1:AMD是预加载，在并行加载js文件同时，还会解析执行该模块（因为还需要执行，所以在加载某个模块前，这个模块的依赖模块需要先加载完成）。CMD是懒加载，虽然会一开始就并行加载js文件，但是不会执行，而是在需要的时候才执行。

2:AMD加载快速，尤其遇到多个大文件，因为并行解析，所以同一时间可以解析多个文件。CMD执行等待时间会叠加。因为每个文件执行时是同步执行（串行执行），因此时间是所有文件解析执行时间之和，尤其在文件较多较大时，这种缺点尤为明显。

3:AMD并行加载，异步处理，加载顺序不一定，可能会造成一些困扰，甚至为程序埋下大坑。CMD因为只有在使用的时候才会解析执行js文件，因此，每个JS文件的执行顺序在代码中是有体现的，是可控的

5:一个网站的js文件过多的情况下，会出现变量名被污染和函数名冲突。有什么解决方法?

1:一个js文件用一个匿名函数自执行。这样一个js文件的变量和函数的作用域只存在这个js文件中 不会出现污染

2:一个js文件一个对象包含着变量和方法，使用的时候去调用这个对象。

3:使用模块化加载js库。比如require.js和sea.js

6:浏览器缓存怎么解决?

1:设置meta标签禁止缓存

```
<!--http-equiv顾名思义，相当于http的文件头作用，
它可以向浏览器传回一些有用的信息，以帮助正确和精确地显示网页内容，
与之对应的属性值为content，content中的内容其实就是各个参数的变量值。-->

//Expires(期限)
//说明：可以用于设定网页的到期时间。一旦网页过期，必须到服务器上重新传输。
<meta   http-equiv="Expires"   CONTENT="0">

//Cache-Control指定请求和响应遵循的缓存机制。no-cache指示请求或响应消息不能缓存
<meta   http-equiv="Cache-Control"   CONTENT="no-cache">

//Pragma(cache模式)
//说明：禁止浏览器从本地计算机的缓存中访问页面内容。
<meta   http-equiv="Pragma"   CONTENT="no-cache">。
```

2:给静态资源文件设置时间戳。每一次请求的文件不一样的话 会去服务器请求最新的资源

```
"www.baidu.com/index.css?timestamp=" + new Date().getTime();
```

3:服务器端设置响应头不缓存

```
//对应html设置meta标签
response.setHeader("Pragma", "no-cache");
response.setHeader("Cache-Control", "no-cache");
response.setDateHeader("Expires", 0);
```

7:说出几条能加快页面加载的方案?

- 减少客户端第一次浏览网站的图片量。使用图片资源懒加载。通过滚动到可视区域去请求图片资源加载。
- 减少http请求 加快速度。css和js和图片的压缩合并
- 把静态资源如js、css、图片都放在cdn上。cdn通过在网络各处放置节点服务器。会按最近服务器推送资源给客户端。
- 服务器端把一些接口可以设置浏览器缓存
- 将脚本放在底部。脚本放在底部对于实际页面加载的时间并不能造成太大影响，但是这会减少页面首屏出现的时间，使页面内容逐步呈现。