性能优化：

```
...
```

跨域：

```
跨域出现的原因：浏览器的同源策略，同源策略会阻止一个域的JavaScript脚本和另外一个域的内容进行交互。
协议：端口号：域名

1.在接口的服务端可以修改响应头   res.header("Access-Control-Allow-Origin","*")
2.jsonp    原理:通过href,src等请求下来的文件是不存在跨域问题的

1、document.domain+iframe的设置

2、动态创建script

3、利用iframe和location.hash

4、window.name实现的跨域数据传输

5、使用HTML5 postMessage

6、利用flash   以上方案见
http://www.cnblogs.com/rainman/archive/2011/02/20/1959325.html#m5

7、nginx反向代理 这个方法一般很少有人提及，但是他可以不用目标服务器配合，不过需要你搭建一个中转nginx服务器，用于转发请求。

8、Jquery JSONP(本质上就是动态创建
script)http://www.cnblogs.com/chopper/archive/2012/03/24/2403945.html
```

1.cookie 和 session 的区别(会话技术)

```
会话技术：
    1.会话：在一次会话中包含多次请求和响应
        一次会话：浏览器第一次给服务器资源发送请求，会话建立，直到有一方断开为止
    2.功能：再一次会话的范围内的多次请求间，共享数据
    3.方式：
        1.客户端会话技术：Cookie
        2.服务器端会话技术：Session
Cookie：
    1.概念：客户端会话技术，将数据存在客户端
    2.使用步骤
        1.创建cookie对象，绑定数据
            var c1=new Cookie(name,value)//name value字符串类型
        2.发送Cookie对象
            response.addCookie(c1)
        3.获取Cookie，拿到数据
            Cookie[]  requset.getCookies()
    3.实现的原理
        基于响应头set-cookie和请求头cookie实现
    4.cookie的细节
```

1.一次可不可以发送多个cookie?
可以
可以创建多个cookie对象，使用response调用多次addCookie方法即可
2.cookie在浏览器的保存时间
默认情况下，浏览器关闭，Cookie数据被销毁
持久化储存：
setMaxAge(seonds)
1.正数：将cookie的数据存到硬盘，并指定过期时间
2.负数：默认值，浏览器关闭就删除
3.0：直接删除cookie信息
3.cookie能不能存中文?
在tomcat 8 之前，cookie中不能直接存中文
需要存中文，一般采用url编码，转码和解码
在tomcat 8 之后，可以存
4.cookie共享问题?
1.假设在同一个服务器上，部署多个web项目，那么这些web项目中cookie能不能共享?
默认状态cookie不可以共享
setPath：可以设置cookie的获取范围，默认状态，就是当前文件的虚拟目录
如果要共享，可以将path设置为"/"
2.不同的服务器间cookie的共享问题?
通过setDomain():如果设置一级域名相同，那么多个服务器之间cookie是可以进行共享的
例：setDomain(".baidu.com"),那么tieba.baidu.com和news.baidu.com中的cookie可以共享，其中一级域名是baidu.com,二级域名是tieba或者news

5.cookie的特点和作用
1.cookie存储数据在客户端浏览器
2.浏览器对于单个cookie的大小有限制(4kb)以及，同一个域名下的总cookie数量也有限制(20个)
作用：
1.cookie一般用于存出少量的不太敏感的数据（不安全，存的比较小）
2.在不登陆的情况下，完成服务器对客户端身份的识别（类似百度首页设置个人偏好，不登陆也可以设置）

session：
1.概念：服务器端会话技术，将数据存在服务器，在一次会话的多次请求之间共享数据
2.使用步骤：
1.获取HttpSession对象：
```
HttpSession session = requset.setSession();
```
2.使用HttpSession对象：
getAttribute()
setAttribute()
removeAttribute()
3.实现原理：
session实现是依赖cookie的。
4.细节：
1.当客户端关闭后，服务器不关闭，两次获取的session是否为同一个?
默认情况下，不是
如果需要相同，则可以创建cookie，键为JSESSIONID，设置最大存活时间，让cookie持久化保存
2.客户端不关闭，服务器关闭后，两次获取的session是同一个吗?
3.session的失效时间?

cookie机制采取的是在客户端保持状态的方案，而session机制采用的是在服务器端保持状态的方案。
会话cookie和持久cookie的区别：
如果不设置过期时间，则表示这个cookie生命周期为浏览器会话期间，只要关闭浏览器窗口,cookie就消失了,会话cookie一般保存在内存里。

如果设置过期时间，浏览器会把cookie保存到硬盘上，关闭后再次打开浏览器，这些cookie依然有效直到超过设定的过期时间。

储存在硬盘的cookie可以在不同的浏览器进程间共享，比如两个ie窗口。而对于保存在内存中的cookie，不同的浏览器有不同的处理方式。

为什么会出现cookie这个机制？
HTTP协议是无状态的协议，一旦数据交换完毕

我的面试题:

1.cookie都有哪些属性?

name: 必填 cookie的名
value: 必填 cookie的值
expire/setMaxAge: 可选 有效时间，三个值，默认负值，浏览器关闭，会话结束，正值，一定时间后关闭，0是删除cookie
Path: 可选 路径，在这个路径以下的页面才可以访问该cookie，可以通过设置该属性来实现一个服务器下，多个项目的cookie共享
Domain: 可选 子域，在这个子域下才可以访问cookie，可以通过设置该属性来实现不同服务器下cookie的共享
Secure: 可选 安全性，指定cookie是否只能通过https协议访问，一般的cookie使用http协议即可访问，如果设置了该属性必须使用https才能访问
HttpOnly: 可选 如果在Cookie中设置了"HttpOnly"属性，那么通过程序（js脚本，Applet等）将无法读取到Cookie的信息，可以有效地防止xss攻击。这个值可以设置为false

组件封装:

高阶组件：connect，接受一个组件作为参数，返回一个新组件

JS实现继承:

实例继承看一下，优缺点，核心内容

深浅拷贝的实现方式:

深拷贝里面的递归看一下，很有可能问

xss攻击和csrf攻击是什么（问安全类的问题如果不太了解，就说了解过这两种攻击方式）：

xss:跨站脚本攻击。
恶意的往网页里插入一段js代码，当用户浏览该页时执行，恶意攻击用户。
危害：窃取cookie，恶意跳转
预防：只要把js的script标签进行一下处理，在提交信息之前做一次转义处理。把<和>转变成&lt和&gt

csrf/xsrf:跨站请求伪造
盗用用户信息，以用户的身份发送恶意请求，包括（发消息，盗账号，购买商品，虚拟货币转账）
登录受信任的网站，并且在本地生成cookie，
在不登出的情况下，访问一个危险的网站就有可能会受到csrf攻击
预防：验证HTTP Referer字段；在请求地址中添加token并验证；在HTTP头中自定义属性并验证。

盒子垂直剧中:

```
div{
    display:flex;
    align-item:center;
    justify-content:center;
},
div{
    position:absolute;
    top:0;
    left:0;
    right:0;
    bottom:0;
    margin:auto;
},
div{
    position:absolute;
    top:50%;
    left:50%;
    margin-left:-width/2;
    margin-top:-top/2;
},
div{
    position:absolute;
    top:50%;
    left:50%;
    transform:translate(-50%,-50%);
}
```

四个盒子水平均匀分布:

```
div{
    display:flex;
    justify-content:space-between;
}
```

小程序的逻辑和视图是不是同一个线程:（双线程）

不是
视图（渲染层）的界面使用了WebView进行渲染
逻辑层采用JsCore线程运行JS脚本

类组件和函数组件的区别:

类组件：类组件中有this指向的问题，而且代码比较多，
函数组件：在react16.8之后新增加的hooks，让函数组件也可以有自己的状态和生命周期，函数组件的性能要比类组件的性能高，因为类组件的使用需要实例化，而函数组件只需要调用函数就可以取到返回值。

用过哪些hooks，usememo是什么（usememo性能优化，问你就提一下）：

Usememo和useCallback的参数和useEffect一致，而且的话，第一个参数是一个函数，第二个参数传一个数组
useMemo和useCallback都会在组件第一次渲染的时候执行，之后会在其依赖的变量发生改变时再次执行
他们和useEffect最大的区别在于后者可以处理一些副作用。并且useMemo返回的是缓存的变量，
useCallback返回的是一个缓存的函数。

hooks 怎么实现componnentDidMount:

在useEffect中的第一个参数初始化的时候只会执行一遍
第二个参数的值发生改变，会再次调用第一个参数，并且当组件被销毁的操作在第一个参数的return的函数中执行。

webapp和小程序有什么区别:

区别：开发一些简单的应用的话，小程序更加的快速，但是小程序提供的这一套开发框架开发复杂应用，比较困难
因为：小程序不支持npm
　　无法复用社区一些成熟的框架
　　组件只能封装view和style，无法封装行为（handler），行为只能定义在page上
　　小程序有1mb的限制，所以的话图片等静态资源要事先放在服务器上
webapp需要开发者自己的host，还需要注册域名甚至备案才能够调用微信接口以及跟公众号集成。
对于简单的应用：小程序合适
稍微复杂一些的应用：webapp更合适
对复杂而且要求性能的应用：native app更合适。

小程序的性能优化:

1.剔除无用的代码逻辑
2.减少代码中的静态资源文件（上传到服务器）
3.复杂逻辑交给服务器端处理之后再返回
4.组件和逻辑复用，减少重复代码
5.分包加载
6.部分页面h5化
7.启用本地缓存
8.数据预加载
9.跳转预加载
10.去掉不必要的事件绑定，去掉不必要的节点属性

component和purecomponent区别:

class com extends React.Component/React.pureComponent
两者基本上差不多
区别：component对数据做深比较，purecomponent对数据做浅比较
相对来说，pure的性能会更好，但是如果数据的层次解构比较复杂，可能会引起数据改变之后组件无法更新的问题。

PureComponent自带通过props和state的浅对比来实现shouldComponentUpadte(),而Component没有。
purecomponent的优点缺点：
不需要开发者自己实现shouldcomponentUpadate，就可以进行简单的判断来提升性能
缺点：可能会因为深层次的数据不一致而产生错误的否定判断，从而shouldComponentUpdate结果返回false，界面得不到更新

性能优化：如果数据更新阶段，不需要在页面展示，可以在shouldComponentUpadate
返回一个false，有利于性能优化。

生命周期：

组件的渲染：
组件将要挂载-render-组件挂载完成-组件是否需要更新-组件将要更新-render-组件更新完成

组件嵌套：
父组件将要挂载-render-子组件将要挂载-子组件render-子组件挂载完成-父组件挂载完成

16.3之后新增的生命周期函数是getDerviedStateFromProps和getSnapshotBeforeUpdate
删除了componentwillreceiverprops

设计模式，具体谈发布订阅怎么实现：

观察者模式：松耦合，观察者与被观察者之间的关系，多用于单个应用内部
订阅者模式：不存在耦合，发布者与订阅者没有必然联系，是通过一个中间的消息站联系起来的，更多的是一种跨应用的模式。

什么是虚拟dom?

虚拟dom是程序员通过js对象模拟的，真实dom在内存中的一种表现形式，虚拟dom和真实的dom是同步的，这个同步的过程就是调和过程。

为什么虚拟dom可以提高性能?

虚拟dom操作的是js对象，然后操作js比操作dom的性能高一些，避免操作dom
虚拟dom树与真正dom树进行对比，比较新旧dom树的差异，从而达到局部渲染，最小化渲染页面，提高性能。

react中key的作用?

为元素添加一个唯一的标识符属性key可以有利于react判断该元素是新创建的还是被移动，或者修改而来的元素，从而最小化实现页面的重新渲染。

调用setState之后发生了什么?

调用setState之后会将传入的参数与当前的状态合并，也就是真实dom和虚拟dom的同步过程，就是调和过程。经过调和过程，react会以高效的方式根据最新的数据状态构建一颗新的dom树，然后重新渲染页面。在react构建新的dom树的过程中，会根据新旧dom的差异进行局部渲染，按需更新。

setState的第二个参数用来做什么?

第二个参数会在`setState`函数调用完成并且组件开始重新渲染的时候调用，一般的话用于监听渲染是否完成