* **使用过的koa2中间件**
* **koa-body原理**
* **介绍自己写过的中间件**
* **有没有涉及到Cluster**
* **介绍pm2**
* **master挂了的话pm2怎么处理**
* **如何和MySQL进行通信**
* **React生命周期及自己的理解**
* **如何配置React-Router**

在v4之前 react-router是集中式管理的 页面中所有的router都在一个文件重定义好了 不允许出现在组件中嵌套的行为。在V4中react-router实现了拆分管理，可以在组件中定于子路由，使得路由的定义更灵活化动态化。比如在V4之前如果两个子路由有公用的模块，只能通过高阶组件和父路由的component 调用props.children来实现。

其他细节：

v4中路由默认不是精确匹配。比如‘/add’可以命中‘/’和‘/add’两个路由。

V4实现精确匹配可以添加exact属性

V4中可以通过添加<Switch>组件只渲染匹配到的第一个路由

V4去除了IndexRouter组件

* **路由的动态加载模块**

路由的动态加载首先就要提到webpack自身的动态加载功能 在webpack中通过import(/\*webpackChunkName:name\*/’filepath’)来实现按需加载，它返回一个promise异步加载模块加载完毕之后变为resolve状态。实际使用中根据不同的框架有不同的调整，在react可以用高阶组件接受异步动态组件，等待组件加载完毕返回动态组件。在Vue中 vue可接受异步组件（一个函数返回import后的promise）

* **服务端渲染SSR**
* **介绍路由的history**

react-router的history:history分为browser、hash、memory三种，分别通过h5的history、url hash值、缓存来处理路由跳转，其中browser利用H5 history特性（可以不刷新的修改页面路径）history.pushState(),history.replaceState()。但pushState和replaceState没有监听事件，无法像hash的onhashchange 无法捕捉事件，更新页面、组件；所以引入了Link这样的内置组件，点击时捕获路径更新对应页面（这也是a标签和Link的不同）；在浏览器的前进后退操作中监听浏览器事件onpopstate，获取history.state，更新页面。

* **介绍Redux数据流的流程**

redux：是一个应用状态管理方案，一般在引用中只会创建一个store对象实例，由createStore()函数创建，该方法返回一个store对象 该对象包含主要的常用方法为：getStore()---获取当前store状态state、dispath()--接受action调用reducer触发store状态更新、subscribe()--设置监听函数，repalaceReducer()--更换reducer;createStore()接受三个参数 ：reducer，初始化状态，和中间件处理函数，也就是常用的applyMiddleware()处理后的中间件，每个中间件会过滤处理一遍action将自己处理完毕后的中间件交给下一个中间件处理，直到成为符合规范的reducer能处理的action。redux的数据流可以整理为 action->中间件->dispatch->reducer

* **Redux如何实现多个组件之间的通信，多个组件使用相同状态如何进行管理**
* **多个组件之间如何拆分各自的state，每块小的组件有自己的状态，它们之间还有一些公共的状态需要维护，如何思考这块**
* **使用过的Redux中间件**

redux-chunk:处理函数类型的中间件

redux-logger:打印日志

* **如何解决跨域的问题**

1:jsonp 利用浏览器js脚本没有跨域限制的原理 将后端接口作为js外链地址插入页面，传递后端接受数据的回调callback函数 操作得到的数据

2：cros跨域 需要后端设置Access-Control-Allow-Origin

3：反向代理

4：postMessage跨域iframe.contentWindow.postMessage(JSON.stringify(data)，'http://www.domain2.com')向iframe发送消息；window.addEventListener(‘message’,()=>{})监听消息；

window.parent.postMessage(JSON.stringify(data), 'http://www.domain1.com');iframe页面发送消息

* **常见Http请求头**
* **移动端适配1px的问题**
* **介绍flex布局**

flex布局也叫弹性布局，在解决了传统css一些基于display、position、float属性的痛点。它分别父元素和子元素。

父元素的属性：flex-direction:定义主轴的方向（默认是横向）

justify-content:子元素在横轴上如何分布（靠右，靠左，居中，两端，等间距）

align-item:定义单排的元素在纵轴上的对其方式（顶端，居中，下端）

align-content:定义多排子元素在纵轴的对其方式（）

flex-wrap:定义一排子元素放不下后是否换行

flex-flow:flex-direction和flex-wrap的组合

子元素的属性：order:排序权重越小越靠前

flex-grow:在排列还有剩余空间时（一行有空隙）设定放大的倍数 且会撑满整个行空间。比如父宽1000，子宽100\*4个，如果设置其中一个的flex-grow:1，那么他将撑满所有剩余空间 宽达到700，其他不变，如果另一个设置了flex-grow为0.25那么子元素宽为160 640 100 100

flex-shrink:如果项目一行放不下 缩小比例，默认为1，放不下都缩小，为0则不会缩小

align-self:设置单个子元素和其他元素不一样的纵轴对其方式 覆盖父元素的align-item属性

* **其他css方式设置垂直居中**

1：设置position:absolute, top,left,right,bottom全都设置为0

2：设置margin 50%，设置子元素的transform:translate(-50%,-50%)

3：父元素设置伪元素 ::after{content:’’,vertical-align:middle,height:100%}

子元素设置vertical-align 中间对其。

* **居中为什么要使用transform（**为什么不使用marginLeft/Top）
* **使用过webpack里面哪些plugin和loader**

loader:

style-loader:将计算好的样式插入页面

css-loader:允许css模块化

less-loader/sass-loader

postcss-loader:css兼容性前缀

babel-loader:bable编译

url-loader:处理图片

ignore-loader:不处理，忽略所引入的文件 比如在node环境中忽略css 忽略img资源

plugins:

html-webpack-plugin:根据模板生成html入口

webpack.HotModuleReplacementPlugin：热模块替换

clean-webpack-plugin：清空文件夹

webpack.optimize.CommonsChunkPlugin：提取公共代码

webpack.DefinePlugin:定义环境变量

extract-text-webpack-plugin：抽取css到单独文件中

* **webpack里面的插件是怎么实现的**
* **dev-server是怎么跑起来**
* **项目优化**
* **抽取公共文件是怎么配置的**

抽取公共代码，在webpack4之前webpack自带了抽取公共代码的插件 webpack.optimize.commonsChunkPlugin，配置公共库名称，抽取缓存runtime。在webpack4中commonChunkPlugin已经被舍弃，有单独的optimize配置项（splitChunk,runtimeChunk,minChunk-最小被共享数，）

* **项目中如何处理安全问题**
* **怎么实现this对象的深拷贝**