1. react生命周期及相关用法

react生命周期分为初始化状态

componentWillMount组件渲染之前

(render渲染组件)

componentDidMount组件渲染之后

运行阶段：

componentWillReceiveProps组件改变时使用

shouldComponentUpdate组件状态改变之前 主要用于手动阻止组件渲染，一般在这个函数中做组件性能优化

componentWillUpdate：组件状态更新前使用

componentDidUpdate组件状态更新后使用

销毁阶段：

componentUnmount:组件销毁阶段。一般用来销毁不用的变量或者解除无用定时器以及解绑无用事件。防止内存泄露问题

1. 运行阶段生命周期调用顺序：

componentReceiveProps-->shouldComponentupdate-->componentWillUpdate-->componentDidUpdate

1. react中key的作用：

key：react中用来追踪列表中哪些元素被修改、删除或者被添加的辅助标识

在diff算法中，key用来判断该元素节点是被移动过来的还是新创建的元素，减少不必要的元素重复渲染。

1. react声明周期中最适合与服务器端进行数据交互的是哪个函数：

componentDidMount:渲染结束后可以进行相关的dom操作

1. react性能优化阶段函数是哪个

shouldComponentUpdate

1. 性能优化方案有哪些：  
   1.重写shouldComponentUpdate来避免不必要的dom操作

2.使用key来帮助列表中子组件的变化

3.使用production版本的react.js

1. react中的组件传值：

父传子：（组件嵌套）父组件定义一个属性，子组件通过this.props接收

子传父：父组件定义一个属性，并将一个回调函数赋值给定义的属性，然后子组件进行调用传过来的函数，并将参数传进去，在父组件的回调函数中即可获得子组件传过来的值。

1. react中的高阶函数

高阶函数就是一个纯js没有副作用的函数

高阶组件就是一个高阶函数，且该组件接收一个组件作为参数返回一个新的组件；（个人：解决两个组件功能类似但是写两个代码繁琐的问题）

1. setState第二个参数作用

setState更改组件的状态是一个异步的过程，所以执行完setState之后不能立刻更改state中的值。第二个参数用来监听state中数据的更改，更改完成后调用回调函数。

详解http协议：http0.9-http1.0-http1.1-http2(2015年发布)

1. OSI（openSystemInterconnection）参考模型或者七层模型：ISO(international organization for standardization)指定的用于计算机或通信系统间互联的标准体系；

TCP/IP分层模型(layening model)：网络接口层--互连网层--传输层--应用层。

绝大多数的web开发都是构建在http协议之上的web应用

特点：支持客户/服务器模式；简单快速（客户向服务器请求服务时只需传送请求方法和路径，通信速度快）；灵活（允许传输任意类型，正在传输的类型由content-type标记）；无连接（每次连接只处理一个请求）无状态（协议对事务处理没有记忆能力）

1. URL：http://host[":"port][abs\_path] ：协议+域名（IP地址）+端口号+请求资源的URI（uniform resource identifier统一资源标识符）

注：http使用统一资源定位符（URI）来传输数据和建立连接。而URL可以认为是一种特殊的URI，包含了用来查找某个资源的足够的信息。但不是所有的URI都是URL，因为URI还包含一个子类：URN（统一资源命名）

URI和URL的区别：

1. 请求request：请求行（request line）请求头部（reader）空行和请求数据组成

* 请求行：请求方法+请求的URI(request-URI)和协议版本GET /562f25980001b1b106000338.jpg HTTP/1.1

有关请求方法：

HTTP1.0定义了三种请求方法： GET, POST 和 HEAD方法。  
HTTP1.1新增了五种请求方法：OPTIONS, PUT, DELETE, TRACE 和 CONNECT 方法。

get:请求指定的页面信息并返回实体主体。

post：向指定资源提交数据进行处理请求，数据放在请求体中。

head：类似于get，但只是请求获取响应信息的报头

put:请求服务器存储一个资源

delete：请求服务器删除一个资源

options：请求查询服务器性能或者查询与资源有关的选项和需求

trace：回显服务器收到的请求主要用于测试和诊断

* 请求头：说明服务器要使用的附加信息

1. 响应response：状态行，消息报头，空行和响应正文

* 状态行：http协议版本号，状态码和状态信息组成

状态行中有状态代码：（三个数字组成，第一个数字定义了响应的类别）

|  |  |
| --- | --- |
| 1xx：指示信息--表示请求已接收，继续处理2xx：成功--表示请求已被成功接收、理解、接受3xx：重定向--要完成请求必须进行更进一步的操作4xx：客户端错误--请求有语法错误或请求无法实现5xx：服务器端错误--服务器未能实现合法的请求 | 200 OK      //客户端请求成功 304 Not Modified 客户的缓存资源是最新的，要客户端使用缓存 400 Bad Request  //客户端请求有语法错误，不能被服务器所理解 401 Unauthorized //请求未经授权，这个状态代码必须和WWW-Authenticate报头域一起使用  403 Forbidden  //服务器收到请求，但是拒绝提供服务 404 Not Found  //请求资源不存在，eg：输入了错误的URL 500 Internal Server Error //服务器发生不可预期的错误 503 Server Unavailable  //服务器当前不能处理客户端的请求，一段时间后可能恢复正常 |

1. 消息报头总结：请求报头+响应的消息报头

|  |  |  |  |
| --- | --- | --- | --- |
| 普通报头 | 请求报头 | 响应报头 | 实体报头 |
| 有少数报头用于所有的请求和响应消息但不用于被传输的实体，只用于传输的消息。eg: | 允许客户端向服务器端传递请求的附加信息以及客户端自身的信息。eg: | 允许服务器传递不能放在状态行中的附加响应信息、有关服务器的信息和对request-URI标识的资源进行下一步访问的信息eg: | 请求和响应信息都可以传送一个实体。一个实体由实体报头和实体正文组成。实体报头定义了实体正文和请求所标识的资源的元信息。 |
| Cache-Control指定缓存指令（单向（响应中出现的缓存指令在请求中未必出现），独立（不影响另一个消息处理的缓存机制）） | Accept:指定客户端接受那些类型的信息  Authorization:证明客户端有权查看某个资源（如果收到响应状态码是401可以发送一个authorization请求报头域的请求要求服务器对其进行验证）  Host:必须；指定请求资源的Internet主机和端口号  User-agent:允许客户端将它的操作系统浏览器和其他属性告诉服务器 | Location:重定向接收者到一个新的位置（常用在更换域名的时候）  Server:包含了服务器用于处理请求的软件信息（与user-agent对应）  WWW-Authenticate:必须包含在401响应信息中；客户端收到401并发送authorization报头请求服务器对其进行验证时，服务器响应报头就包含该报头域 | Content-Type：指明发送给接收者的实体正文的媒体类型  Content-Type:text/html;charset=ISO-8859-1  Content-Encoding:媒体类型的修饰符，上面媒体类型的解码机制  Expires:响应过期日期和时间 |

1. 工作原理：

客户端连接web服务器建立TCP套接字连接--发送http请求--服务器接收请求返回HTTP响应---释放TCP连接---客户端浏览器解析HTML内容进行渲染

1. 相关技术补充：
2. 基础：高层协议：域名系统服务DNS（domain name system）；文件传输协议FTP（file transfer protocol）
3. 中介：代理（proxy）网关（gateway）通道（tunnel）

* 代理：
* 网关
* 通道

TCP协议详解：TCP（transmission control protocol传输控制协议）

TCP/IP不是一个协议而是一个协议族的统称，里面包含了IP协议、IMCP协议和TCP协议，UDP协议以及http,pop3协议等等。

|  |  |
| --- | --- |
|  | 应用层：向用户提供一组常用的应用程序  传输层：提供应用程序之间的通信  网络层：负责相邻计算机之间的通信  网络接口层：负责接收IP（Internet protocol）数据并通过网络发送 |

TCP/IP意味着TCP和IP一起协同工作。TCP负责应用软件（比如browser）和网络软件之间的通信，IP负责计算机之间的通信。

SYN同步标志ACK确认标志 FIN结束标志

**TCP三次握手（three-way handshake）**

指建立一个TCP连接需要客户端和服务端总共发送三个包以确认连接的建立。

即：建立连接时，客户端发送SYN到服务器并进入SYN-SEND状态等待服务器确认；服务器收到SYN包后发送给客户端SYN+ACK包，服务器进入SYN-RECV状态；

客户端收到服务器的SYN+ACK包，向服务器发送确认ACK包完成三次握手，客户端和服务器开始传送数据。

**TCP四次挥手（four-way wavehand）**

即断开一个TCP连接时，需要客户端和服务端总共发送四个包确认连接的断开。

客户端发送一个FIN用来关闭客户端到服务端的数据传送；服务端收到FIN后发送一个ACK给客户端；服务端发送一个FIN用来关闭服务端到客户端的数据发送；客户端收到FIN后发送一个ACK给服务端，完成四次挥手；

**为什么建立连接是三次握手，关闭连接是四次挥手？**

因为服务端收到客户端建立连接SYN报文后把ACK和SYN一起发送给客户端；

而关闭连接时，收到客户端的FIN后仅仅表示客户端不再向服务端发送数据了但是还能接收数据，服务端可能发送一些数据给客户端后再发送FIN给客户端统一关闭连接；

**TCP和UDP的区别**

UDP user data protocol用户数据报协议

非连接协议--不需要维护连接状态，信息包标题很短，尽最大努力交付，面向报文。

我们经常用ping命令来测试两台主机之间的TCP/IP通信是否正常，原理就是向对方主机发送UDP数据包如果对方主机确认收到数据包且数据包是否到达的消息及时反馈回来，那么网络就是通的

区别：

1.基于连接还是无连接

2.对系统资源要求（TCP较多 UDP较少）

3.UDP程序结构较简单

4.TCP保证数据正确性；UDP可能丢包；TCP保证顺序性，UDP不保证

5.TCP流模式 UDP数据报模式

流模式和数据报模式：

通过TCP连接给另一端发送数据100个字节，接收方可以分10次接收每次10字节。

通过UDP发送：发送端调用了几次write接收端必须用相同次数的read读完；对报文不拆分也不合并；

**当我们在浏览器的地址栏输入网址，然后回车：**

域名解析--发起TCP的三次握手--建立起TCP连接后发起http请求---服务器响应http请求得到HTML代码--浏览器解析HTML代码--页面渲染

**如何提高页面加载速度**

1.减少http请求（图片地图 css雪碧图，特别是图片比较多的网站如果能用css sprites降低图片数量带来的将是速度的提升）

2.使用CDN

3.添加expires头

4.压缩组件

5.将样式表放在头部，希望页面尽快显示内容，为用户提供可视化的反馈

6.将脚本放在底部。因为js的下载和执行会阻塞DOM树的构建（严谨说是中断了dom树的更新）

7.使用外部的js和CSS，当脚本或样式是从外部引入的文件，浏览器有可能缓存它们，从而以后加载的时候能够直接使用缓存；

8.减少DNS查找

9.精简js和css，删除重复脚本

10.避免重定向

重定向：用于用户从一个URL重新路由到另一个URL

常用重定向的类型：

301：永久重定向（主要用于网站的域名发生变更后）

302：临时重定向（主要实现post请求后告知浏览器转移到新的URL）

304：not modified（主要用于当浏览器在其缓存中保留了组件的一个副本，同时组件已经过期了，浏览器就会生成一个get请求，如果服务器的组件没有被修改过就会返回304状态码，同时不携带主体，告知浏览器可以重用这个副本）

损伤性能：页面重定向后，延迟整个HTML文档的传输，在HTML文档到达之前页面不会显示任何东西，也没有任何组件被下载；

重定向的应用场景：跟踪内部流量：当拥有一个门户主页时，同时想对用户离开主页后的流量进行跟踪，这时可以使用重定向；跟踪出站流量（?）

11.在进行ajax请求的时候选择尽量使用get方法，这样可以使用客户端的缓存，提高请求速度。（因为post的请求是不可以在客户端缓存的，每次请求都需要发送给服务器进行处理）