博彦面试准备 20220303

1. 从昨天下午接收到贵公司的hr面试通知后，我大部分时间都在准备react。您可以问我一些react的问题。
2. 两个框架不同点：

vue是在html里面写js，react跟他相反是在js里面写html。

vue用于小型项目，react用于大型项目。

1. react组件封装方式：vue（分为局部和全局注册 传递）

函数组件 rfc react function component 通过es5构造函数结构创建，一般以数据的展示为主，功能简单，组件中的逻辑代码较少

类组件 rcc react class component 通过es6的类构造，允许使用更多功能，如组件状态数据，声明周期钩子，访问redux仓库等

1. react优缺点：

优点：

（1）可以通过构造函数或者类结构描述视图组件，

（2）集成虚拟DOM（渲染性能好）

（3）单项数据流（更容易追踪数据变化排查问题）

（4）一切都是compont；代码更加模块化，重用代码更容易，可维护性更高

（5）大量使用es6新特性

缺点;

jsx的一个问题是，渲染函数render（）尝尝包含大量逻辑，最终看这更像是程序片段，而不是视觉呈现。后期如果发生需求更改，维护起来比较麻烦。

1. 编写方式：

脚本引入方式：script src 属性

react.js react对象管理react虚拟DOM

react-dom.js ReactDOM对象，渲染操作DOM的

babel.js 转移js语法 转移jsx语法为js代码

脚手架方式：

create-react-app

1. 脚手架配置。

node环境: node-v 检查node版本要求在14版本以上

npm -v 检查npm 是否存在（npm config get reqistry）

修改为国内地址：

npm config set reqistry <http://reqistry.npm.taobao.ory/>

1. 生成项目包

npx 临时安装软件的相关依赖命令，使用完毕之后会删除

create-react-app 脚手架

reactpro 项目包名称 可以自定义

npx create-react-app reactpro

1. 进入目录启动服务

cd reactpro

npm start

1. 生命周期

componentWillMount（） 渲染前

componentDidMount（）渲染后

componentWillReceiveProps（）接受props传值时

shouldComponentUpdate（）控制组件是否更新 这个函数是用来判断是否需要调用render函 数重新描绘DOM, 因为DOM的绘制非常消耗性能, 如果我能能在这个函数中写一些优化算法逻辑,控制DOM绘制的频率和次数, 则能极大的提高网页渲染效率, 优化性能

componentWillUpdate（）组件将要更新

componentDidUpdata（）组件已经更新

componentWillunmount（）组件将要移除

1. 访问流程

浏览器中输入 localhost:3000 localhost本地域名 3000服务占用的端口

react脚手架启动3000端口服务

react启动的服务默认访问的是public目录下的 index.html

在 index.html 中找到 <div id="root"></div> 挂在点

webpack运行会自动打包 src/index.js 到 index.html 里

src/index.js 引入了 src/App.js 一个函数组件

1. react组件中 通过state管理组件内部状态类似于vue中的data
2. setState舒心DOM，具有异步性。
3. 样式使用：标签属性style行内样式，属性classs链接外部样式。
4. 双向绑定：

在表单元素中，单项数据流：数据渲染到页面DOM上。

双向数据流：既满足单项数据流，又可以接收到用户输入信息。

1. 事件触发：this指向问题！

严格模式下undefined react默认严格模式

1. 遍历不修改源数据 foreach

map处理数组的每个元素返回新数组

在react中会把数组的元素挨个渲染到页面上

添加渲染标签的时候，返回一个数组

15.view跟组件，view组件的默认布局就是flex，主轴竖向排列横向拉伸

**押题**

1. 原型原型链：js中的对象都有一个属性叫做ptoto 也是一个对象。表示对象的原型。当访问对象中的属性或者方法时，首先在对象本身中寻找，如果找不到则会在原型中寻找，原型中也找不到就会在原型的原型中找，知道最顶层。

js中的类（构造函数）都有一个prototype的属性，表示蓓蕾的原型类型，通过这个类实例化的对象（这个构造函数创建的对象），proto指向本类的prototype，从而实现了类方法的共享。一个类的prototype也是一个对象，他也有proto，把他的proto指向另一个累的prototype时，那么这个类的对象就能访问另一个类中的方法，从而实现了方法的继承。

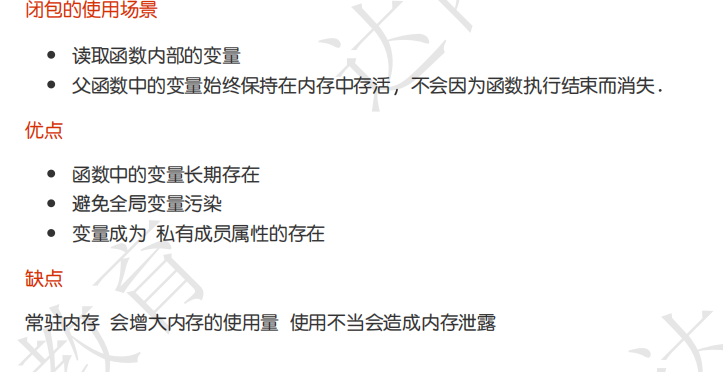
A类的prototype.proto指向B类 ，Bprototype又可以指向C这种结构就叫原型链。

1. 闭包

当一个函数A的作用域被内部的B函数引用时，A函数的作用域就会被B函数闭包，当A函数执行完毕，A函数作用域也不会释放。

闭包可以实现对象的私有属性和私有方法。

闭包可以封装变量，从而减少对全局作用域的污染。



1. 防抖，节流。

防抖：对于频繁触发的事件，如果只希望最后一次或者第一执行。则需要使用防抖。 搜索框

节流:对于频繁触发的事件，希望按照一定的频率进行绑定函数调用，则使用函数节流。鼠标点击 监听滚动事件

都可以settimeout实现。

1. es6新特性。

箭头函数（箭头函数内部this跟函数所在上下文this保持一致；没有arguments参数几个，可以用rest替代；不能使用call bind apply来改变this指向）

字符串模板，let块级作用域提升，const常量，class声明类

结构赋值，promise（处理异步）,es6模块化（react模块化比vue更完整）

1. vue与react区别

相同点：

都支持服务器渲染

都有虚拟dom，组件化开发，通过props参数进行父子组件数据的传递

都有webcomponent规范

都是数据驱动视图

都有状态管理 react有redux ；vue有vuex

不同点：

react严格上只针对mvc的view层，vue的mvvm模式

虚拟dom不一样，vue会跟中每一组件的依赖关系，不需要重新渲染整个dom树。而react不同，当应用的状态被改变时，全部组件都会重现渲染，所以react中用sholdcomponentUpdate这个生命周期钩子函数来控制

组件写法不一样

数据绑定不一样，vue实现了数据双向绑定，react数据流动是单向的在react中，state对象需要用setstate方法更新状态，在vue中，sate对象不是必须的数据由data属性在vue对象中管理。

1. 盒子模型

标准盒子模型和怪异盒子模型

**标准盒模型的width和height属性的范围只包含了content，**

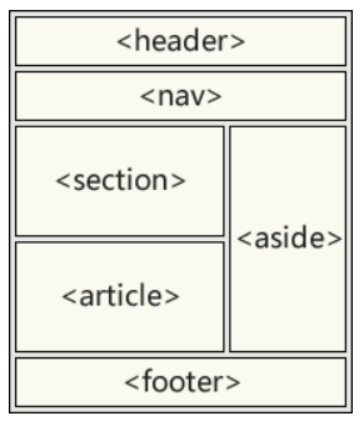
**IE盒模型的width和height属性的范围包含了border、padding和content。**

可以通过修改元素的box-sizing属性来改变元素的盒模型：

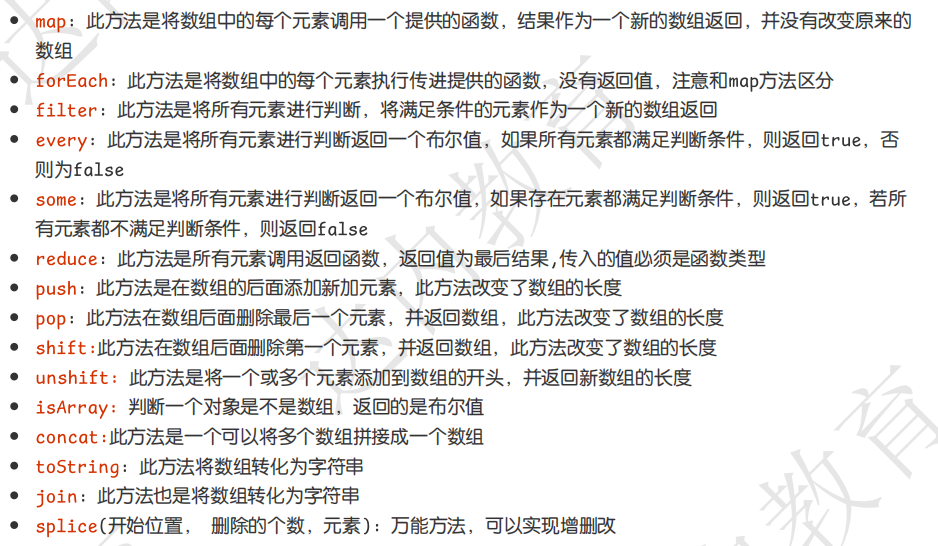
表示标准盒模型（默认值）

表示IE盒模型（怪异盒模型）

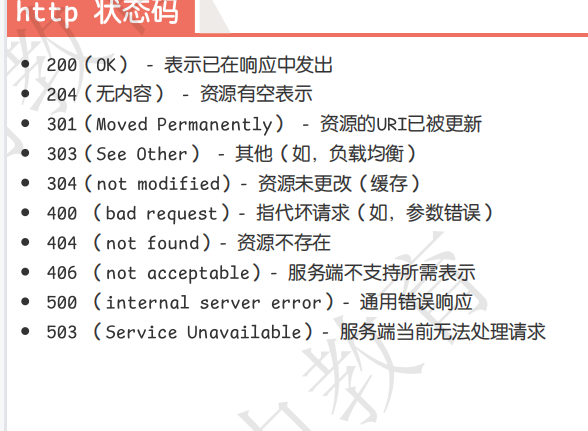
1. 继承的6种方式
2. 原型链继承 原型对象的数据会同时被修改
3. 构造函数继承 借助call 不会继承方法
4. 组合继承
5. 原型式继承 实例的引用类型属性指向相同的内存
6. 寄生式继承
7. 寄生式组合继承
8. js 特性 封装 继承 多态

html5语义化标签

1. 数组常用方法：



10.http状态码



1. 水平垂直居中

一：定宽高类：

1. 绝对定位和负magin值
2. 绝对定位+tranform
3. 绝对定位+left/right/bottom/top+margin
4. flex布局
5. grid布局

二：不定宽高

（1）绝对定位+transform

（2）table-cell

（3）flex布局

扩展：

内联元素居中：

1. 水平居中

* 行内元素可设置：text-align: center;
* flex布局设置父元素：display: flex; justify-content: center;

1. 垂直居中

* 单行文本父元素确认高度：height === line-height
* 多行文本父元素确认高度：disaply: table-cell; vertical-align: middle;

块级元素居中：

1. 水平居中

* 定宽: margin: 0 auto;
* 不定宽： 参考上诉例子中不定宽高例子。

1. 垂直居中

* position: absolute设置left、top、margin-left、margin-to(定高)；
* position: fixed设置margin: auto(定高)；
* display: table-cell；
* transform: translate(x, y)；
* flex(不定高，不定宽)；
* grid(不定高，不定宽)，兼容性相对比较差；

11.js执行环境

全局环境

script标签就是全局执行环境

函数环境

当一个函数被调用时，在函数内部就形成一个函数执行环境

eval()

把字符串当做js代码执行

12.执行上下文：

创建阶段 ：创建变量对象 建立作用域链 确定this指向

执行阶段：主要完成变量赋值 函数调用和一些其他操作