## React

1. setState到底是异步还是同步?

答：有时是异步的，有时是同步的。

* 1. setState只在合成事件和钩子函数中是“异步”的，在原生事件和setTimeout中都是同步的
  2. setState的“异步”并不是说内部由异步代码实现，其实本身执行的过程和代码都是同步的，只是合成事件和钩子函数的调用顺序在更新之前，导致在合成事件和钩子函数中没法立马拿到更新后的值，形成了所谓的“异步”，当然可以通过第二个参数setState(partialState,callback)中的callback拿到更新后的结果。
  3. setState的批量更新优化也是建立在“异步”（合成事件、钩子函数）之上的，在原生事件和setTimeout中不会批量更新，在“异步”中如果对同一个值进行多次setState，setState的批量更新策略会对其进行覆盖，取最后一次的执行，如果是同时setState多个不同的值，在更新时会对其进行合并批量更新。

1. 触发多次setState,那么render会执行几次？
   1. 多次执行setState会合并为一次render,因为setState并不会立即改变state的值，而是将其放到一个任务队列里，最终将多个setState合并，一次性更新页面。所以我们可以在代码里多次调用setState，每次只需要关注当前修改的字段即可
2. 调用setState之后发生了什么？
   1. 在代码中调用setState函数之后，React会将传入的参数对象与组件当前的状态合并，然后触发所谓的调和过程。
   2. 经过调和过程，React会以相对高效的方式根据新的状态构建React元素树并且着手重新渲染整个UI界面。
   3. 在React得到元素树之后，React 会自动计算出新的树与老树的节点差异，然后根据差异对界面进行最小化重新渲染。
   4. 在差异计算算法中，React 能够相对精确的知道哪些位置发生了改变以及应该如何改变，这就保证了按需更新，而不是全部重新渲染。
3. 父组件调用子组件的方法
4. React 懒加载

## WEB 安全

1. 网页验证码是干嘛的，是为了解决什么安全问题?
   1. 网页的图片验证码是用于人机识别的，用于区分人的操作行为和机器行为，防止恶意机器盗刷，恶意占票。
   2. 网页的登录验证码，属于双因素认证应用到账号安全的范畴，作用是确保是用户本人进行登录，大大降低了由于账号被盗，导致的非授权登录行为。
   3. 网页的注册验证码，是验证注册者的身份，防止恶意注册，确保用户的有效性。

## JS

1. 节流和防抖 ?
   1. 防抖函数:就是指触发事件后在n秒内，函数只能执行一次，如果在n秒内又触发了事件，则会重新计算函数执行时间;如果你一直猛触发 就不给你执行了
   2. 节流函数:就是指连续触发事件但是在一段时间中只执行一次函数
2. 事件委托是什么？

事件委托本质上是利用了浏览器事件冒泡的机制。因为事件在冒泡过程中会上传到父节点，并且父节点可以通过事件对象获取到目标节点，因此可以把节点的监听函数定义在父节点上，由父节点的监听函数统一处理多个子元素的事件，这种方式称为事件代理。 使用事件代理我们可以不必要为每一个子元素都绑定一个监听事件，这样减少了内存上的消耗，并且使用事件代理我们还可以实现事件的动态绑定，比如说新增了一个子节点，我们并不需要单独得为他添加一个监听事件，他所发生的事件会交给父元素中的监听函数来处理。

1. JS有几种方法判断变量的类型？

0. 数据类型

1. 6个简单的数据类型（原始类型）

1. String

2. Number

3. Boolean

4. Undefined

5. Null

6. Symbol

2. 一个复杂的数据类型 Object

1. 使用 typeof 操作符 检测

1. 对原始值很有用，但对引用值没什么用

2. typeof 是操作符不是函数，因此没有参数

eg. typeof 1 // number

2. 使用 instanceof 检测

1. 检测具体类型的对象

eg. colors instanceof Array // 变量是Array吗？

3. 使用constructor检测

4. Object.prototype.toString.call(val).slice(8, -1).toLowerCase();

const typeCheck = (val) => {

let type = Object.prototype.toString.call(val).slice(8, -1).toLowerCase();

return type;

};