问题?

Immutable

https://www.cnblogs.com/yoissee/p/6001465.html
https://www.jianshu.com/p/0fa8c7456c15

Immutable.is 比较的是两个对象的 hashCode 或 valueOf（对于 JavaScript 对象）。由于 immutable 内部使用了 Trie 数据结构来存储，只要两个对象的 hashCode 相等，值就是一样的。这样的算法避免了深度遍历比较，性能非常好。
普通的 Mutable 对象的深拷贝操作会将一整份数据都复制一遍，而 Immutable 对象在修改数据时并不会复制一整份数据，而是将变化的节点与未变化的节点的父子关系转移到一个新节点上，类似于链表的结构。从 “复制” 的角度来看，做到了最小化的复制，未变化的部分都是共享的，Mutable 在复制的时候是 “全量”，而 Immutable 复制的是 “增量”，对于内存空间的使用率的比较高低立判。

Redux-thunk
允许dispatch 返回一个函数

```
export const getHomeInfo = () => {
    return (dispatch) => {
        axios.get('/api/home.json').then((res) => {
            const result = res.data.data;
            dispatch(changHomeData(result));
        });
    }
}
```

1 dispatch(fund)

```
fun = ()=>{
      //异步
      axois.().then(
            dispatch(data)
      )
}
```

2 applyMiddleware

```
    dispatch = compose(...chain)(store.dispatch)
```

3 creatthunkMiddleWare

```
 if (typeof action === 'function') {
                return action(dispatch, getState, extraArgument);
            }
```

返回函数的action creator

设计模式
观察者
https://www.cnblogs.com/leaf930814/p/9014200.html

错误监控：
即时运行错误：
1）try catch 2）window.onerror

资源加载错误：
1）object.onerror [img script]

2）performance.getEntries（）
3）window.addEventListener('error',()=>,true) 事件捕获

上报错误：
1）Ajax
2）Image 对象上报 src="http://baidu.com?test=XXX"

DNS预解析

JSONP

window[callbackName]注册回调名称
创建sciptt标签
script.onLoad state 删除script callBack = null
script.onLoad.onerror = ()=>{}
发送： document.getElementsByTagName('head')[0].appendChild(script)

CORS
origin
Access-Control-Allow-Origin: http://api.bob.com
Access-Control-Allow-Methods: GET, POST, PUT

```
// CORS 【参考资料】(http://www.ruanyifeng.com/blog/2016/04/cors.html)
// url (必须), options (可选)
fetch('/some/url', {
    method: 'get',
}).then(function (response) {

}).catch(function (err) {
    // 出错了;等价于 then 的第二个参数,但这样更好用更直观
});
```

postMessage

A->B
A iframe

```
<iframe id="iframe" src="http://target.com/target.html"></iframe>
<input id="msg" type="text" placeholder="请输入要发送的消息">
<button id="send">发送</button>
```

iframeWindow.postMessage(msg,"http://target.com/target.html");

B接受，监听addelision（”message”，event[origin,data]）{

}

XMLHttpRequest
new XML
xhr.open (GET<POST<url,)
xhr.send(null,data)
xhr.onLoad=(){
xhr.status ===200 304 206
}

```
var xhr = new XMLHttpRequest();

    xhr.onreadystatechange = function() {
        if (xhr.readyState == 4 && xhr.status == 200) {
            alert( xhr.responseText );
        }
    };
```

```
        xhr.open('get', 'https://api.douban.com/v2/book/search?q=javascript&count=1',
true);
        xhr.send();
```

40分钟看完js主流面试题
https://www.jianshu.com/p/49ad676a19e6?utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation

13 Promise

https://www.jianshu.com/p/8d3bd325e886

https://segmentfault.com/a/1190000009478377

14 Axois
https://www.jianshu.com/p/93fa30986d07

15 CORS

11 Batch_updates_transactoins
setState 异步更新
https://www.cnblogs.com/danceonbeat/p/6993674.html

9、前端性能监控
https://blog.csdn.net/bxl0218/article/details/80813699

五分钟撸一个前端性能监控工具
http://web.jobbole.com/94938/

10、js继承

12、RN原理
jsbridge ocbrgidge

messagequeue

moduleId , methodId, args

callbacks

1、原生请求取消

```
var native = new XMLHttpRequest();
  native.open("GET","https://api.github.com/");
  native.send();
  native.onreadystatechange=function(){
      if(native.readyState==4&&native.status==200){
          console.log(native.response);
      }else{
          console.log(native.status);
      }
  }
native.abort();
```

Axois 请求取消 source, custom 分别是CancelToken.source() 【cancelToken】 的标识

```
var CancelToken = axios.CancelToken;

var source = CancelToken.source();
axios({
    method:"GET",
    url:"https://api.github.com/",
    cancelToken:source.token
}).then((res) => {
    console.log(res.data);
}).catch((err) => {
    console.log(err);
});

var custom = CancelToken.source();
axios({
    method:"GET",
    url:"https://api.github.com/",
    cancelToken:custom.token
}).then((res)=>{
    console.log(res.data);
}).catch((err)=>{
    console.log(err);
});

source.cancel('Operation canceled by the user.');
custom.cancel('精确取消');
```

2-1、函数防抖

延时要执行的代码

```
const debouce = (func, wait)=> {
    let timer;
    return ()=>{
        clearTimeout(timer);
        timer = setTimeout(func, wait);
    }
}
document.getElementById("debounce").onscroll= debouce(()=>console.log(1) ,900);
```

2-2、函数节流

一段时间内 只能执行一次

3、**将URL的参数以对象形式返回**

4、COOKIE安全与防护
https://blog.csdn.net/cheeseandcake/article/details/72794749

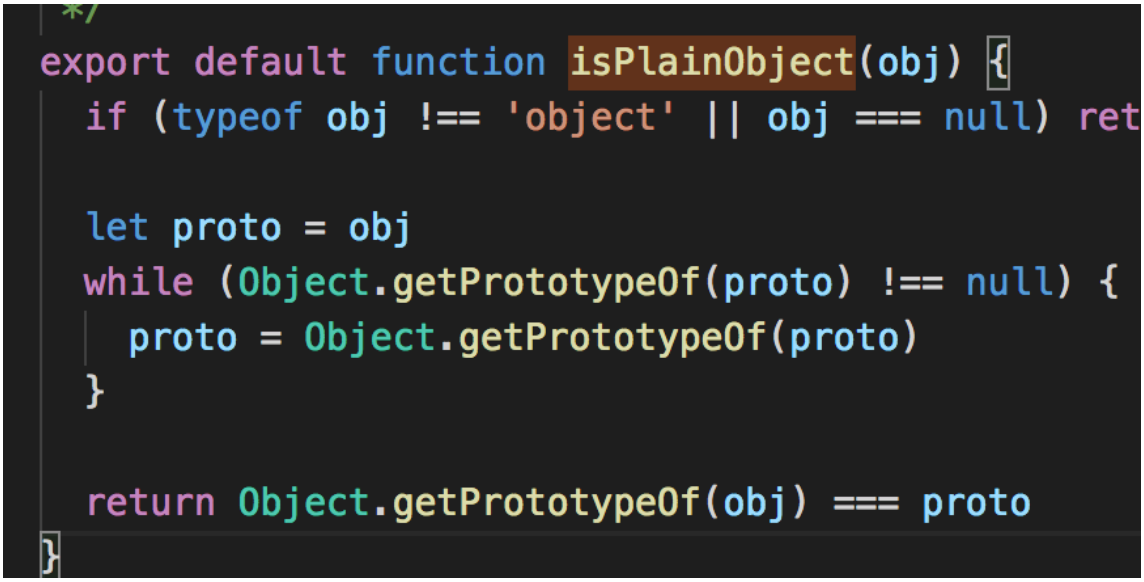
expires=； null==> session cookie
Max-age=;
Domian=www.exaple.com;
Path=/;
Secure : https;
HttpOnly: 只有浏览器发出 HTTP 请求时

5、Diffing 算法

https://www.jianshu.com/p/3ba0822018cf
O(n3)—O(n)
策略一（**tree diff**）：
Web UI中DOM节点跨层级的移动操作特别少，可以忽略不计。
策略二（**component diff**）：
拥有相同类的两个组件 生成相似的树形结构，
拥有不同类的两个组件 生成不同的树形结构。
策略三（**element diff**）：
对于同一层级的一组子节点，通过唯一id区分。

6、redux
redux源码解析:
https://www.cnblogs.com/yangyangxxb/p/10047648.html

https://www.jianshu.com/p/728dbb1de25e
isPlainObject(obj)



```
*/
export default function isPlainObject(obj) {
  if (typeof obj !== 'object' || obj === null) ret

  let proto = obj
  while (Object.getPrototypeOf(proto) !== null) {
    proto = Object.getPrototypeOf(proto)
  }

  return Object.getPrototypeOf(obj) === proto
}
```

是普通对象 new Object() 所创建的对象了.
getState()
subscribe() 返回一个unsubscribe()方法
dispatch(action)

observable()

7、加密方式

对称加密：

非对称 安全性好 慢

Hash算法

8、 手动实现Promise all race

https://blog.csdn.net/qq_41047322/article/details/82670596

https://blog.csdn.net/weixin_38858002/article/details/82380700

https://www.jianshu.com/p/f3413db4f64f