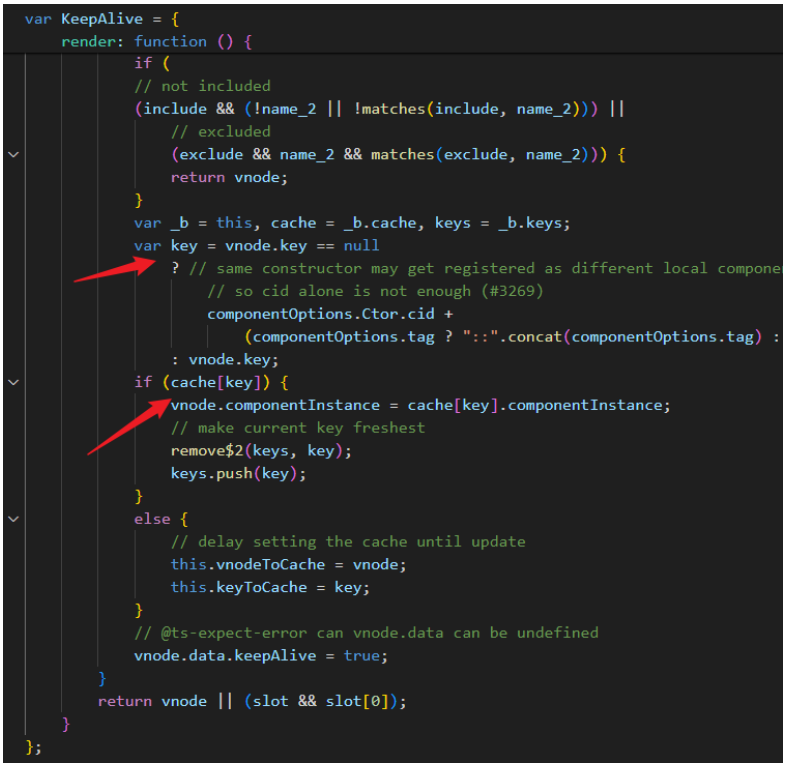
keep-alive组件bug

- 场景：
  - 新建标签功能中，即需要保存组件状态，也得保证新建的标签与旧的是独立的，那么需要用到key值绑定。
- 问题：
  - 当在keep-alive中，router-view上添加key，在本地开发中触发vue热重载会导致页面白屏，控制台中调试可以发现组件未被渲染。
- 过程：
  - 可以看到keep-alive组件中取缓存组件是通过key进行存储取值的。



```
var KeepAlive = {
    render: function () {
        if (
        // not included
        (include && (!name_2 || !matches(include, name_2))) ||
            // excluded
            (exclude && name_2 && matches(exclude, name_2))) {
            return vnode;
        }
        var _b = this, cache = _b.cache, keys = _b.keys;
        var key = vnode.key == null
            ? // same constructor may get registered as different local compone
                // so cid alone is not enough (#3269)
                componentOptions.Ctor.cid +
                    (componentOptions.tag ? "::".concat(componentOptions.tag) :
            : vnode.key;
        if (cache[key]) {
            vnode.componentInstance = cache[key].componentInstance;
            // make current key freshest
            remove$2(keys, key);
            keys.push(key);
        }
        else {
            // delay setting the cache until update
            this.vnodeToCache = vnode;
            this.keyToCache = key;
        }
        // @ts-expect-error can vnode.data can be undefined
        vnode.data.keepAlive = true;
    }
    return vnode || (slot && slot[0]);
}
};
```

  - 1.存在key值：
    - 存在Key那么触发热更新时，重新渲染router-view组件，由于前后key值是不会改变的，所以他可以取cache中缓存的vnode，将缓存的vnode赋值到componentInstance上，后续更新组件需要经过patch过程更新组件，需要渲染vnode为真实dom，然后将componentInstance.$el赋值给vnode.elm上（initComponets）,之后通过insert去插入到父节点里，直接使用的是dom方法appendChild，由于当前页面中已经存在了相同引用的节点，所以appendChild插入不了的，patch更新的最后需要去删除旧节点，那么这个旧节点其实就是我们现在页面上的dom了，所以执行完删除旧dom，页面白屏。
      - patch
        - 会进行sameVnode判断，由于热更新重新渲染了router-view组件，会重新创建vnode，这个时候tag值就发生变化，所以sameVnode判断为

false，走不到patchVnode更新，而是走外层 删除旧节点，创建新节点逻辑

```
    }
    return function patch(oldVnode, vnode, hydrating, removeOnly) {
        if (isUndef(vnode)) {
            if (isDef(oldVnode))
                invokeDestroyHook(oldVnode);
            return;
        }
        var isInitialPatch = false;
        var insertedVnodeQueue = [];
        if (isUndef(oldVnode)) {
            // empty mount (likely as component), create new root element
            isInitialPatch = true;
            createElm(vnode, insertedVnodeQueue);
        }
        else {
            var isRealElement = isDef(oldVnode.nodeType);
            if (!isRealElement && sameVnode(oldVnode, vnode)) {
                // patch existing root node
                patchVnode(oldVnode, vnode, insertedVnodeQueue, null, null, removeOnly);
            }
            else {
                if (isRealElement) {
                    // mounting to a real element
                    // check if this is server-rendered content and if we can perform
                    // a successful hydration.
                    if (oldVnode.nodeType === 1 && oldVnode.hasAttribute(SSR_ATTR)) {
```

```
var hooks = ['create', 'activate', 'update', 'remove', 'destroy'];
function sameVnode(a, b) {
    return (a.key === b.key &&
        a.asyncFactory === b.asyncFactory &&
        ((a.tag === b.tag &&
            a.isComment === b.isComment &&
            isDef(a.data) === isDef(b.data) &&
            sameInputType(a, b)) ||
            (isTrue(a.isAsyncPlaceholder) && isUndef(b.asyncFactory.error))));
}
function sameInputType(a, b) {
```

- createComponet

```
    }
    function createComponent(vnode, insertedVnodeQueue, parentElm, refElm) {
        var i = vnode.data;
        if (isDef(i)) {
            var isReactivated = isDef(vnode.componentInstance) && i.keepAlive;
            if (isDef((i = i.hook)) && isDef((i = i.init))) {
                i(vnode, false /* hydrating */);
            }
            // after calling the init hook, if the vnode is a child component
            // it should've created a child instance and mounted it. the child
            // component also has set the placeholder vnode's elm.
            // in that case we can just return the element and be done.
            if (isDef(vnode.componentInstance)) {
                initComponent(vnode, insertedVnodeQueue);
                insert(parentElm, vnode.elm, refElm);
                if (isTrue(isReactivated)) {
                    reactivateComponent(vnode, insertedVnodeQueue, parentElm, ref
                }
                return true;
            }
        }
    }
    function initComponent(vnode, insertedVnodeQueue) {
```

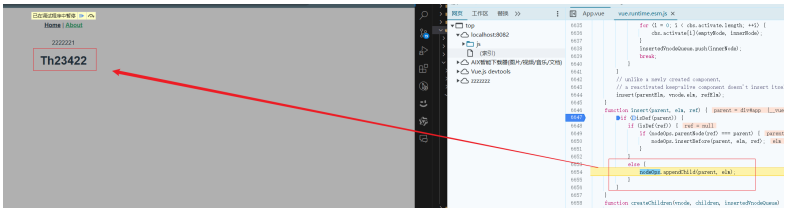
  - initComponent：赋值vnode的真实节点

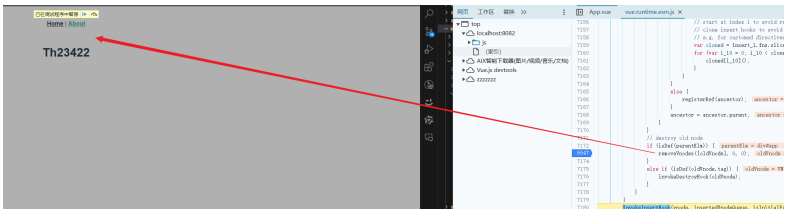
```
function initComponent(vnode, insertedVnodeQueue) {
    if (isDef(vnode.data.pendingInsert)) {
        insertedVnodeQueue.push.apply(insertedVnodeQueue, vnode.data.pendingInsert);
        vnode.data.pendingInsert = null;
    }
    vnode.elm = vnode.componentInstance.$el;
    if (isPatchable(vnode)) {
        invokeCreateHooks(vnode, insertedVnodeQueue);
        setScope(vnode);
    }
    else {
        // empty component root.
        // skip all element-related modules except for ref (#3455)
        registerRef(vnode);
        // make sure to invoke the insert hook
        insertedVnodeQueue.push(vnode);
    }
}
```

  - insert

```
}
function insert(parent, elm, ref) {
    if (isDef(parent)) {
        if (isDef(ref)) {
            if (nodeOps.parentNode(ref) === parent) {
                nodeOps.insertBefore(parent, elm, ref);
            }
        }
        else {
            nodeOps.appendChild(parent, elm);
        }
    }
}
```

- 2.不存在key值：
  - 可以看到key值的赋值，如果不存在key，那么会取cid拼接tag作为key，这个cid在每次vue重载组件后都会进行更新++操作，所以每当vue重载后，key都会更新，所以就不存在上述的情况。
- 3.单纯切换页面：
  - 上诉过程一致，插入新节点，然后删除旧节点





- 原因：
  - 触发vue热重载时重新渲染组件，前后key值不发生变化，componentInstance取缓存组件的值，tag值发生改变，sameVnode判断失效，导致触发patch创建新节点删除旧节点逻辑，那这个新节点取的又是componentInstance上的el，引用与旧节点是一样的，导致appendChild添加不进去，最后删除旧节点，导致组件消失，页面空白。
- 解决：
  - 多加一层cid的判断，如果cid相同说明没有进行vue热重载可以用缓存vnode，如果不同说明已经进行了热重载，那么就不能取缓存vnode。

```
if (cache[key]) {
    //判断cid是否相同，不同则有过热重载的reload，需要重建缓存
    if (vnode._cid === cache[key]._cid) {
        vnode.componentInstance = cache[key].componentInstance;
        // make current key freshest
        remove(keys, key);
        keys.push(key);
    } else {
        cache[key].componentInstance.$destroy();
        cache[key] = vnode
    }

} else {
```

以上内容整理于 幕布文档