Virtual DOM

课程目标

课程目标

- 了解什么是虚拟 DOM，以及虚拟 DOM 的作用
- Snabbdom 的基本使用
- Snabbdom 的源码解析

Virtual DOM

什么是 Virtual DOM

什么是 Virtual DOM

- Virtual DOM(虚拟 DOM)，是由普通的 JS 对象来描述 DOM 对象
- 真实 DOM 成员

```
let element = document.querySelector('#app')
let s = ''
for (var key in element) {
  s += key + ','
}
console.log(s)
```

真实 DOM 成员

```
align,title,lang,translate,dir,hidden,accessKey,draggable,spellcheck,autocapitalize,contentEditable,isContentEditable,inputMode,offsetParent,offsetTop,offsetLeft,offsetWidth,offsetHeight,style,innerText,outerText,oncopy,oncut,onpaste,onabort,onblur,oncancel,oncanplay,oncanplaythrough,onchange,onclick,onclose,oncontextmenu,oncuechange,ondblclick,ondrag,ondragend,ondragenter,ondragleave,ondragover,ondragstart,ondrop,ondurationchange,onemptied,onended,onerror,onfocus,oninput,oninvalid,onkeydown,onkeypress,onkeyup,onload,onloadeddata,onloadedmetadata,onloadstart,onmousedown,onmouseenter,onmouseleave,onmousemove,onmouseout,onmouseover,onmouseup,onmousewheel,onpause,onplay,onplaying,onprogress,onratechange,onreset,onresize,onscroll,onseeked,onseeking,onselect,onstalled,onsubmit,onsuspend,ontimeupdate,ontoggle,onvolumechange,onwaiting,onwheel,onauxclick,ongotpointercapture,onlostpointercapture,onpointerdown,onpointermove,onpointerup,onpointercancel,onpointerover,onpointerout,onpointerenter,onpointerleave,onselectstart,onselectionchange,onanimationend,onanimationiteration,onanimationstart,ontransitionend,dataset,nonce,autofocus,tabIndex,click,focus,blur,enterKeyHint,onformdata,onpointerrawupdate,attachInternals,namespaceURI,prefix,localName,tagName,id,className,classList,slot,part,attributes,shadowRoot,assignedSlot,innerHTML,outerHTML,scrollTop,scrollLeft,scrollWidth,scrollHeight,clientTop,clientLeft,clientWidth,clientHeight,attributeStyleMap,onbeforecopy,onbeforecut,onbeforepaste,onsearch,elementTiming,previousElementSibling,nextElementSibling,children,firstElementChild,lastElementChild,childElementCount,onfullscreenchange,onfullscreenerror,onwebkitfullscreenchange,onwebkitfullscreenerror,setPointerCapture,releasePointerCapture,hasPointerCapture,hasAttributes,getAttributeNames,getAttribute,getAttributeNS,setAttribute,setAttributeNS,remov
```

真实 DOM 成员

```
ren,firstElementChild,lastElementChild,childElementCount,onfullscreenchange,onfullscreenerror,onwe
bkitfullscreenchange,onwebkitfullscreenerror,setPointerCapture,releasePointerCapture,hasPointerCap
ture,hasAttributes,getAttributeNames,getAttribute,getAttributeNS,setAttribute,setAttributeNS,remov
eAttribute,removeAttributeNS,hasAttribute,hasAttributeNS,toggleAttribute,getAttributeNode,getAttri
buteNodeNS,setAttributeNode,setAttributeNodeNS,removeAttributeNode,closest,matches,webkitMatchesSe
lector,attachShadow,getElementsByTagName,getElementsByTagNameNS,getElementsByClassName,insertAdjac
entElement,insertAdjacentText,insertAdjacentHTML,requestPointerLock,getClientRects,getBoundingClie
ntRect,scrollIntoView,scroll,scrollTo,scrollBy,scrollIntoViewIfNeeded,animate,computedStyleMap,bef
ore,after,replaceWith,remove,prepend,append,querySelector,querySelectorAll,requestFullscreen,webki
tRequestFullScreen,webkitRequestFullscreen,createShadowRoot,getDestinationInsertionPoints,ELEMENT_
NODE,ATTRIBUTE_NODE,TEXT_NODE,CDATA_SECTION_NODE,ENTITY_REFERENCE_NODE,ENTITY_NODE,PROCESSING_INST
RUCTION_NODE,COMMENT_NODE,DOCUMENT_NODE,DOCUMENT_TYPE_NODE,DOCUMENT_FRAGMENT_NODE,NOTATION_NODE,DO
CUMENT_POSITION_DISCONNECTED,DOCUMENT_POSITION_PRECEDING,DOCUMENT_POSITION_FOLLOWING,DOCUMENT_POSI
TION_CONTAINS,DOCUMENT_POSITION_CONTAINED_BY,DOCUMENT_POSITION_IMPLEMENTATION_SPECIFIC,nodeType,no
deName,baseURI,isConnected,ownerDocument,parentNode,parentElement,childNodes,firstChild,lastChild,
previousSibling,nextSibling,nodeValue,textContent,hasChildNodes,getRootNode,normalize,cloneNode,is
EqualNode,isSameNode,compareDocumentPosition,contains,lookupPrefix,lookupNamespaceURI,isDefaultNam
espace,insertBefore,appendChild,replaceChild,removeChild,addEventListener,removeEventListener,disp
atchEvent
```

Virtual DOM

- 使用 Virtual DOM 来描述真实 DOM

```
{
  sel: "div",
  data: {},
  children: undefined,
  text: "Hello Virtual DOM",
  elm: undefined,
  key: undefined
}
```

Virtual DOM

为什么要使用 Virtual DOM

为什么要使用 Virtual DOM

- 前端开发刀耕火种的时代
- MVVM 框架解决视图和状态同步问题
- 模板引擎可以简化视图操作，没办法跟踪状态
- 虚拟 DOM 跟踪状态变化
- 参考 github 上 virtual-dom 的动机描述
  - 虚拟 DOM 可以维护程序的状态，跟踪上一次的状态
  - 通过比较前后两次状态差异更新真实 DOM

案例演示

- jQuery-demo
- Snabbdom-demo

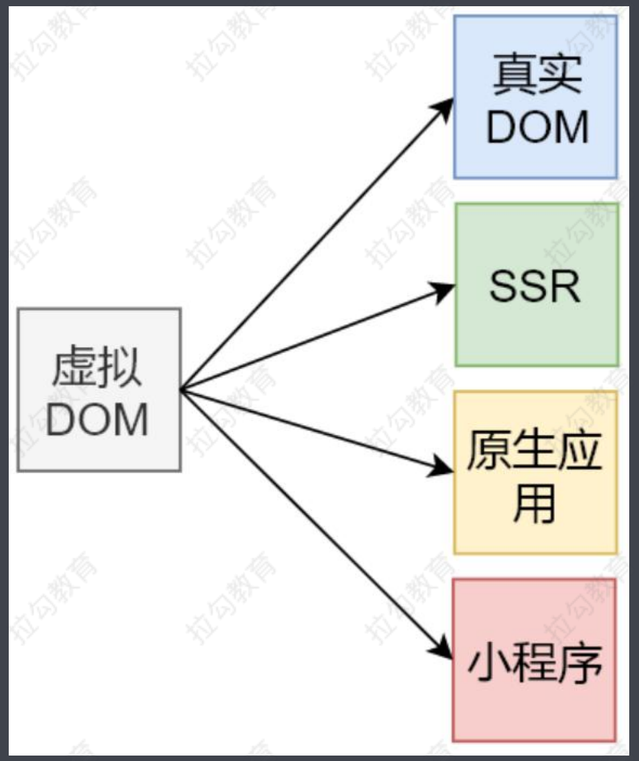
Virtual DOM

虚拟 DOM 的作用和虚拟 DOM 库

虚拟 DOM 的作用

- 维护视图和状态的关系
- 复杂视图情况下提升渲染性能
- 跨平台
  - 浏览器平台渲染DOM
  - 服务端渲染 SSR(Nuxt.js/Next.js)
  - 原生应用(Weex/React Native)
  - 小程序(mpvue/uni-app)等

虚拟 DOM 的作用



虚拟 DOM 库

- Snabbdom
  - Vue.js 2.x 内部使用的虚拟 DOM 就是改造的 Snabbdom
  - 大约 200 SLOC（single line of code）
  - 通过模块可扩展
  - 源码使用 TypeScript 开发
  - 最快的 Virtual DOM 之一
- virtual-dom

Snabbdom 基本使用

创建项目

步骤

STEPS

- 安装 parcel
- 配置 scripts
- 目录结构

安装 parcel

```
# 创建项目目录
md snabbdom-demo
# 进入项目目录
cd snabbdom-demo
# 创建 package.json
npm init -y
# 本地安装 parcel
npm install parcel-bundler -D
```

配置 scripts

```
"scripts": {
  "dev": "parcel index.html --open",
  "build": "parcel build index.html"
}
```

目录结构

Snabbdom 基本使用

导入 Snabbdom

Snabbdom 文档

- 看文档的意义
  - 学习任何一个库都要先看文档
  - 通过文档了解库的作用
  - 看文档中提供的示例，自己快速实现一个 demo
  - 通过文档查看 API 的使用
- Snabbdom 文档
  - https://github.com/snabbdom/snabbdom
  - 当前版本 v2.1.0

导入 Snabbdom

- 安装 Snabbdom
  - npm intall snabbdom@2.1.0
- 导入 Snabbdom
  - Snabbdom 的两个核心函数 init 和 h()
    - init() 是一个高阶函数，返回 patch()
    - h() 返回虚拟节点 VNode，这个函数我们在使用 Vue.js 的时候见过

导入 Snabbdom

- 文档中导入的方式

```
import { init } from 'snabbdom/init'
import { h } from 'snabbdom/h'
const patch = init([])
```

- 实际导入的方式
  - parcel/webpack 4 不支持 package.json 中的 exports 字段

```
import { h } from 'snabbdom/build/package/h'
import { init } from 'snabbdom/build/package/init'
```

Snabbdom 基本使用

案例1

Snabbdom 基本使用

案例2

Snabbdom 基本使用

Snabbdom 中的模块

模块

Module

- 模块的作用
- 官方提供的模块
- 模块的使用步骤

模块的作用

- Snabbdom 的核心库并不能处理 DOM 元素的属性/样式/事件等，可以通过注册 Snabbdom 默认提供的模块来实现
- Snabbdom 中的模块可以用来扩展 Snabbdom的功能
- Snabbdom 中的模块的实现是通过注册全局的钩子函数来实现的

官方提供的模块

- attributes
- props
- dataset
- class
- style
- eventlisteners

模块使用步骤

- 导入需要的模块
- init() 中注册模块
- h() 函数的第二个参数处使用模块

Snabbdom 源码解析

概述

如何学习源码

- 宏观了解
- 带着目标看源码
- 看源码的过程要不求甚解
- 调试
- 参考资料

Snabbdom 的核心

- init() 设置模块，创建 patch() 函数
- 使用 h() 函数创建 JavaScript 对象(VNode)描述真实 DOM
- patch() 比较新旧两个 Vnode
- 把变化的内容更新到真实 DOM 树

Snabbdom 源码

- 源码地址
  - https://github.com/snabbdom/snabbdom
  - 当前版本：v2.1.0
- 克隆代码
  - git clone -b v2.1.0 --depth=1 https://github.com/snabbdom/snabbdom.git

Snabbdom 源码解析

h 函数

h 函数介绍

- 作用：创建 VNode 对象
- Vue 中的 h 函数

```
new Vue({
  router,
  store,
  render: h ⇒ h(App)
}).$mount('#app')
```

- h 函数最早见于 hyperscript，使用 JavaScript 创建超文本

函数重载

- 参数个数或参数类型不同的函数
- JavaScript 中没有重载的概念
- TypeScript 中有重载，不过重载的实现还是通过代码调整参数

函数重载-参数个数

```ts
function add (a: number, b: number) {
  console.log(a + b)
}
function add (a: number, b: number, c: number) {
  console.log(a + b + c)
}
add(1, 2)
add(1, 2, 3)
```

函数重载-参数类型

```
function add (a: number, b: number) {
  console.log(a + b)
}
function add (a: number, b: string) {
  console.log(a + b)
}
add(1, 2)
add(1, '2')
```

Snabbdom 源码解析

常用快捷键

Snabbdom 源码解析

VNode

Snabbdom 源码解析

patch 整体过程分析

patch 整体过程分析

- patch(oldVnode, newVnode)
- 把新节点中变化的内容渲染到真实 DOM，最后返回新节点作为下一次处理的旧节点
- 对比新旧 VNode 是否相同节点(节点的 key 和 sel 相同)
- 如果不是相同节点，删除之前的内容，重新渲染
- 如果是相同节点，再判断新的 VNode 是否有 text，如果有并且和 oldVnode 的 text 不同，直接更新文本内容
- 如果新的 VNode 有 children，判断子节点是否有变化，

Snabbdom 源码解析

init

Snabbdom 源码解析

patch

Snabbdom 源码解析

调试 patch 函数

Snabbdom 源码解析

createElm

Snabbdom 源码解析

调试 createElm

Snabbdom 源码解析

removeVnodes 和 addVnodes

拉勾教育
—互联网人实战大学—

Snabbdom 源码解析

patchVnode

Snabbdom 源码解析

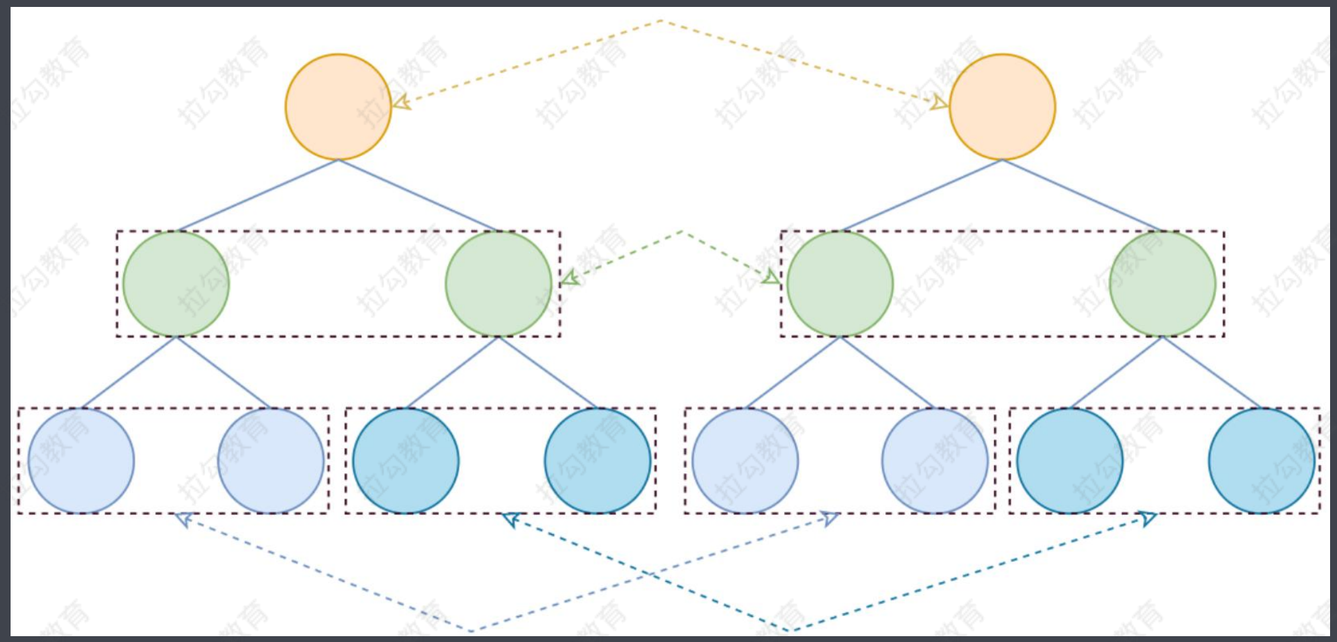
updateChildren 整体分析

Diff 算法

- 虚拟 DOM 中的 Diff 算法
  - 查找两颗树每一个节点的差异

Diff 算法

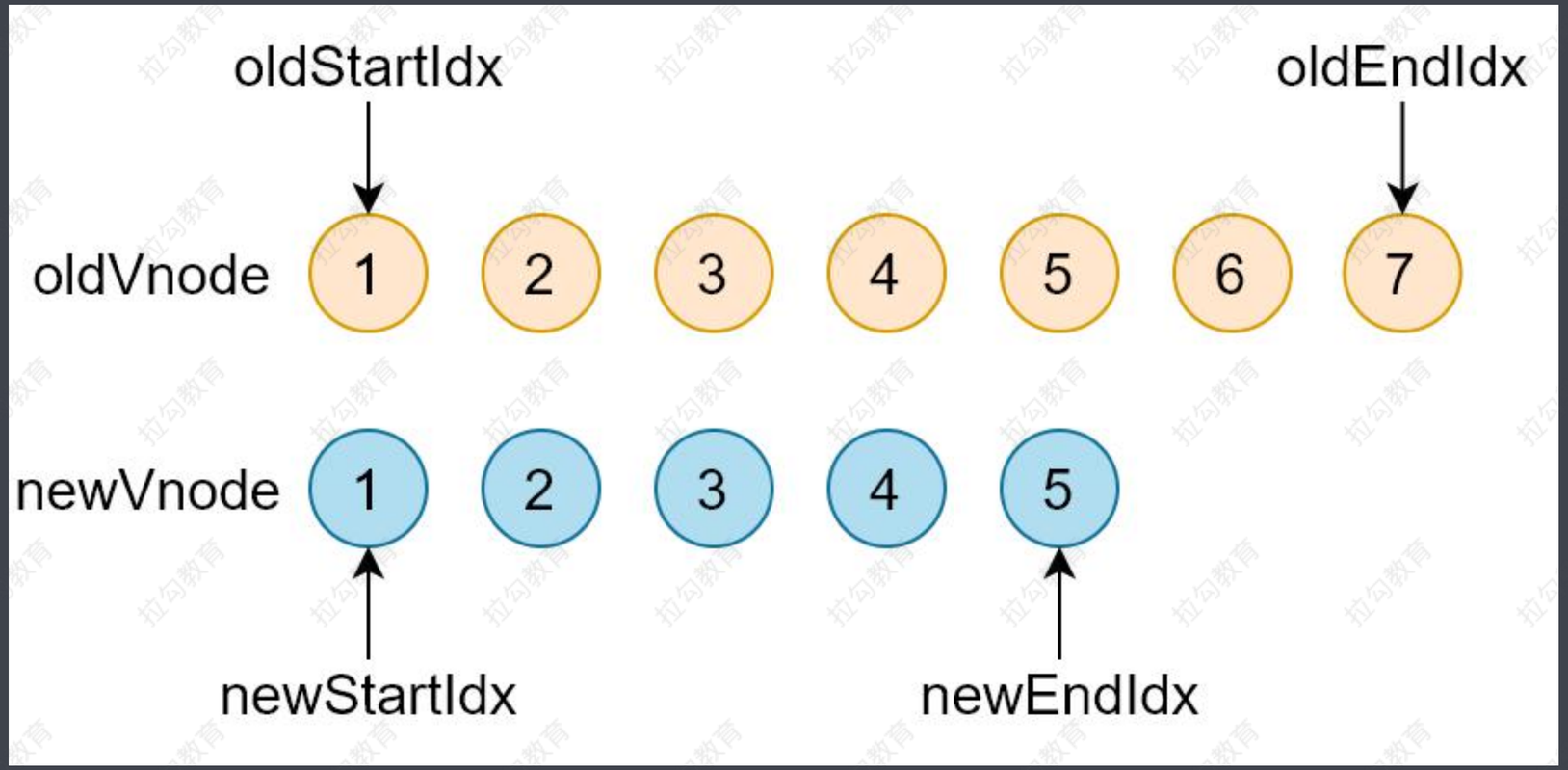
- Snbbdom 根据 DOM 的特点对传统的diff算法做了优化
  - DOM 操作时候很少会跨级别操作节点
  - 只比较同级别的节点



执行过程

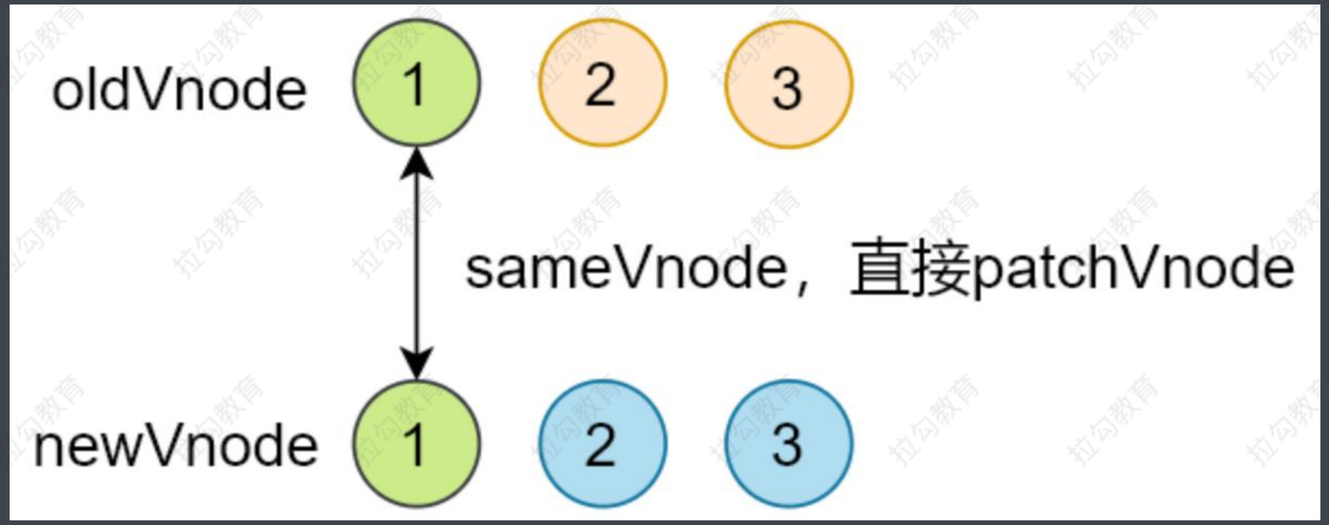
- 在对开始和结束节点比较的时候，总共有四种情
  - oldStartVnode / newStartVnode（旧开始节点 / 新开始节点）
  - oldEndVnode / newEndVnode（旧结束节点 / 新结束节点）
  - oldStartVnode / oldEndVnode（旧开始节点 / 新结束节点）
  - oldEndVnode / newStartVnode（旧结束节点 / 新开始节点）

执行过程


oldStartIdx
oldEndIdx
oldVnode
1
2
3
4
5
6
7
newVnode
1
2
3
4
5
newStartIdx
newEndIdx

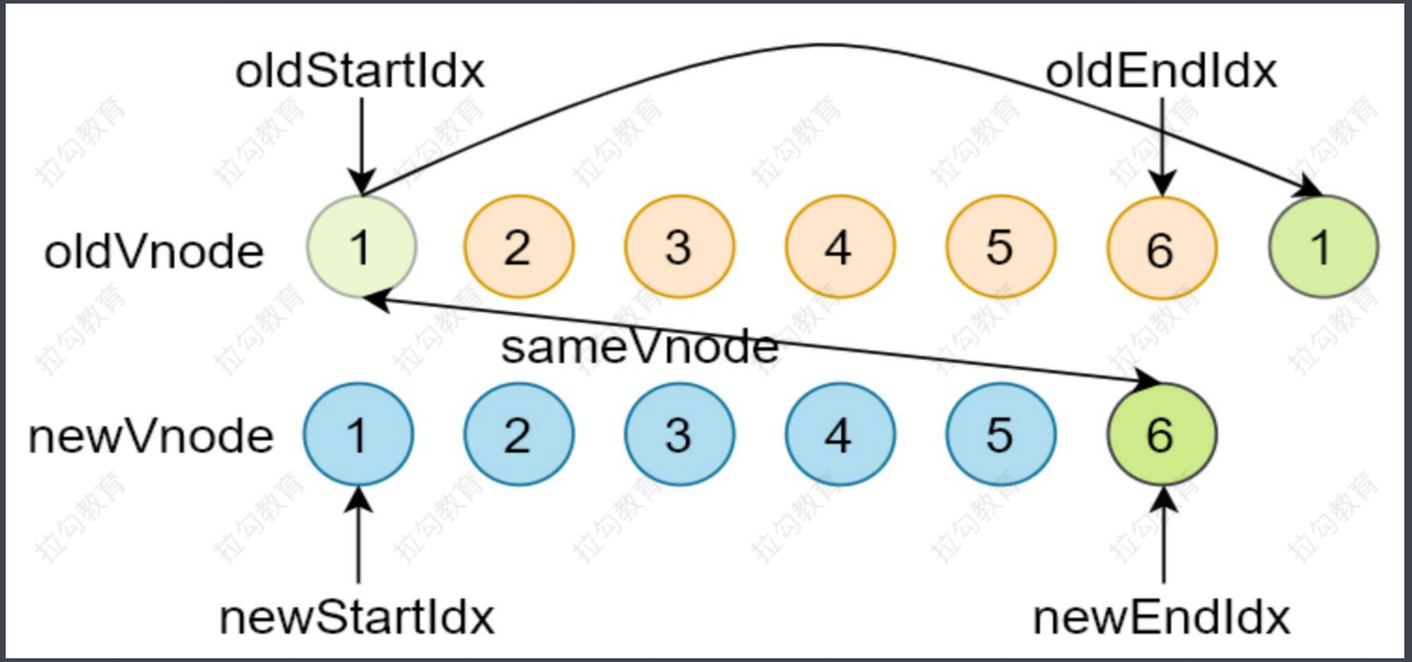
开始和结束节点

- 如果新旧开始节点是 sameVnode（key 和 sel 相同）
  - 调用 patchVnode() 对比和更新节点
  - 把旧开始和新开始索引往后移动 oldStartIdx++ / oldEndIdx++



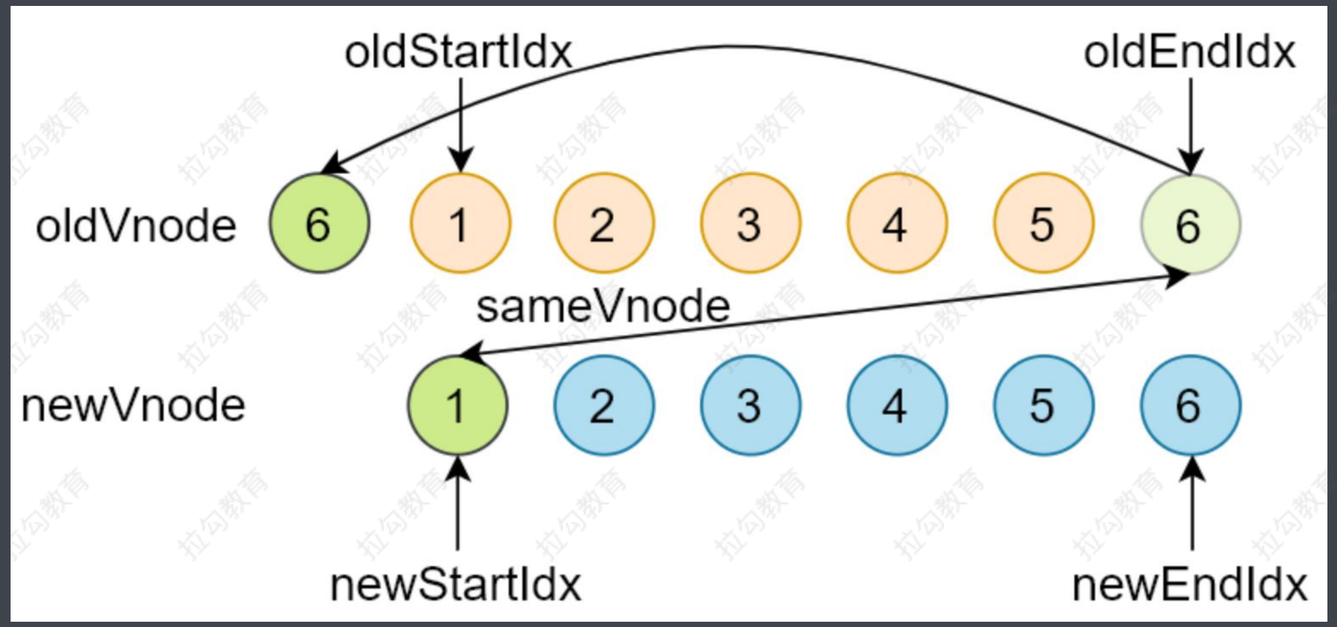
旧开始节点 / 新结束节点

- 调用 patchVnode() 对比和更新节点
- 把 oldStartVnode 对应的 DOM 元素，移动到右边，更新索引

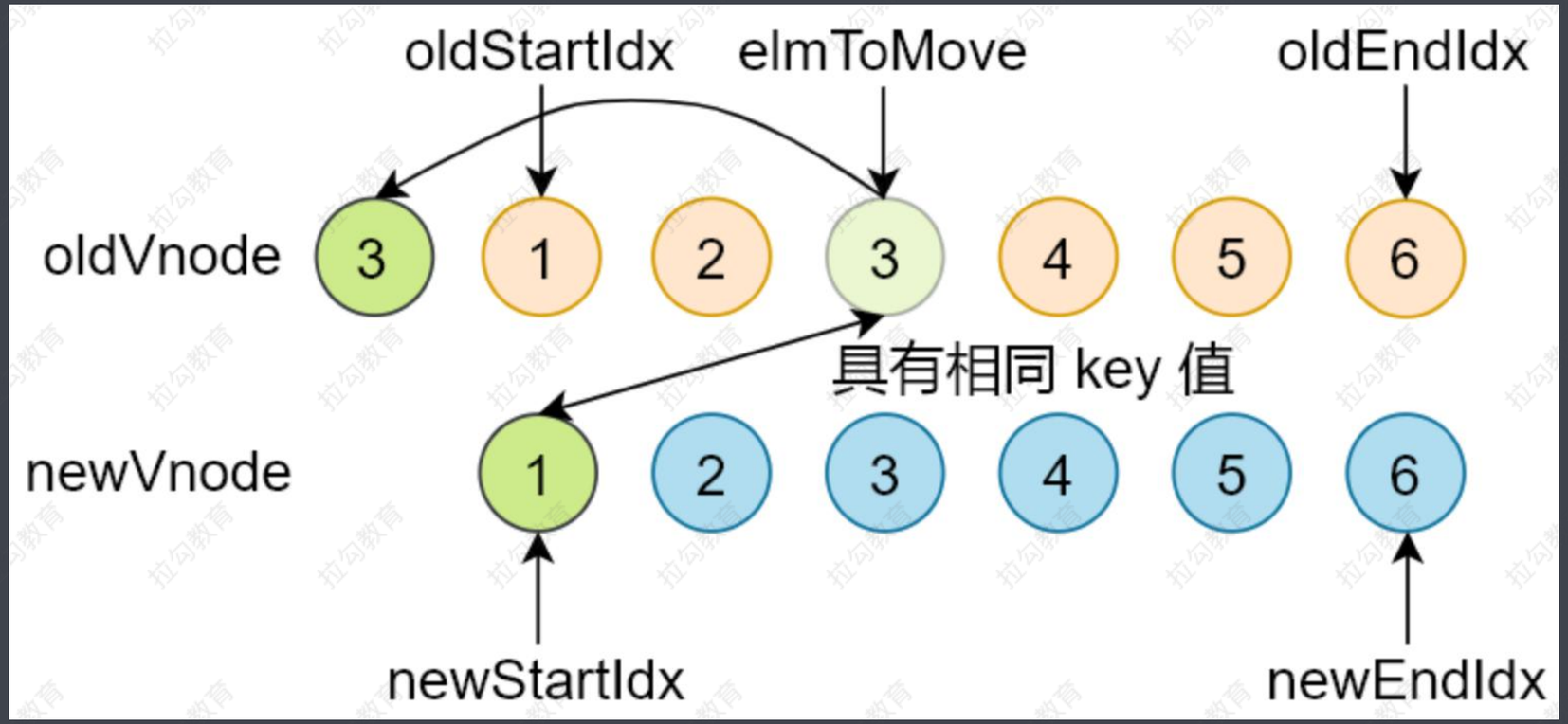


旧结束节点 / 新开始节点

- 调用 patchVnode() 对比和更新节点
- 把 oldEndVnode 对应的 DOM 元素，移动到左边，更新索引



非上述四种情况


oldStartIdx
elmToMove
oldEndIdx
oldVnode
3
1
2
3
4
5
6
具有相同 key 值
newVnode
1
2
3
4
5
6
newStartIdx
newEndIdx

非上述四种情况

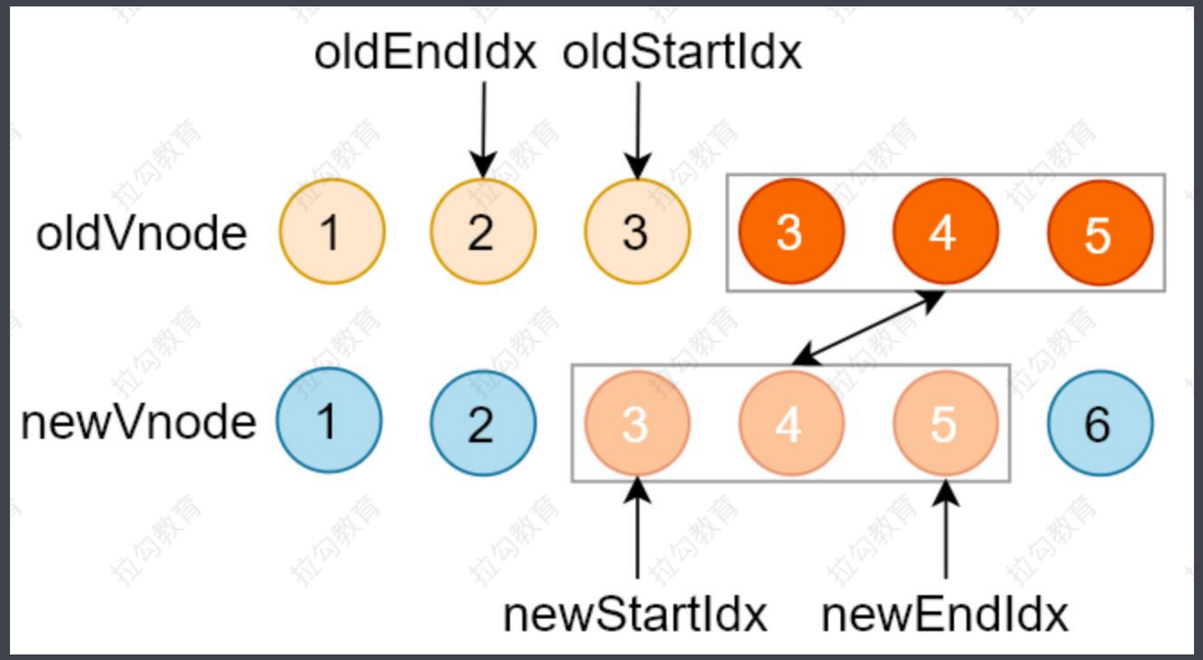
- 遍历新节点，使用 newStartNode 的 key 在老节点数组中找相同节点
- 如果没有找到，说明 newStartNode 是新节点
  - 创建新节点对应的 DOM 元素，插入到 DOM 树中
- 如果找到了
  - 判断新节点和找到的老节点的 sel 选择器是否相同
  - 如果不相同，说明节点被修改了
    - 重新创建对应的 DOM 元素，插入到 DOM 树中
  - 如果相同，把 elmToMove 对应的 DOM 元素，移动到左边

循环结束

- 当老节点的所有子节点先遍历完（oldStartIdx > oldEndIdx），循环结束
- 新节点的所有子节点先遍历完（newStartIdx > newEndIdx），循环结束

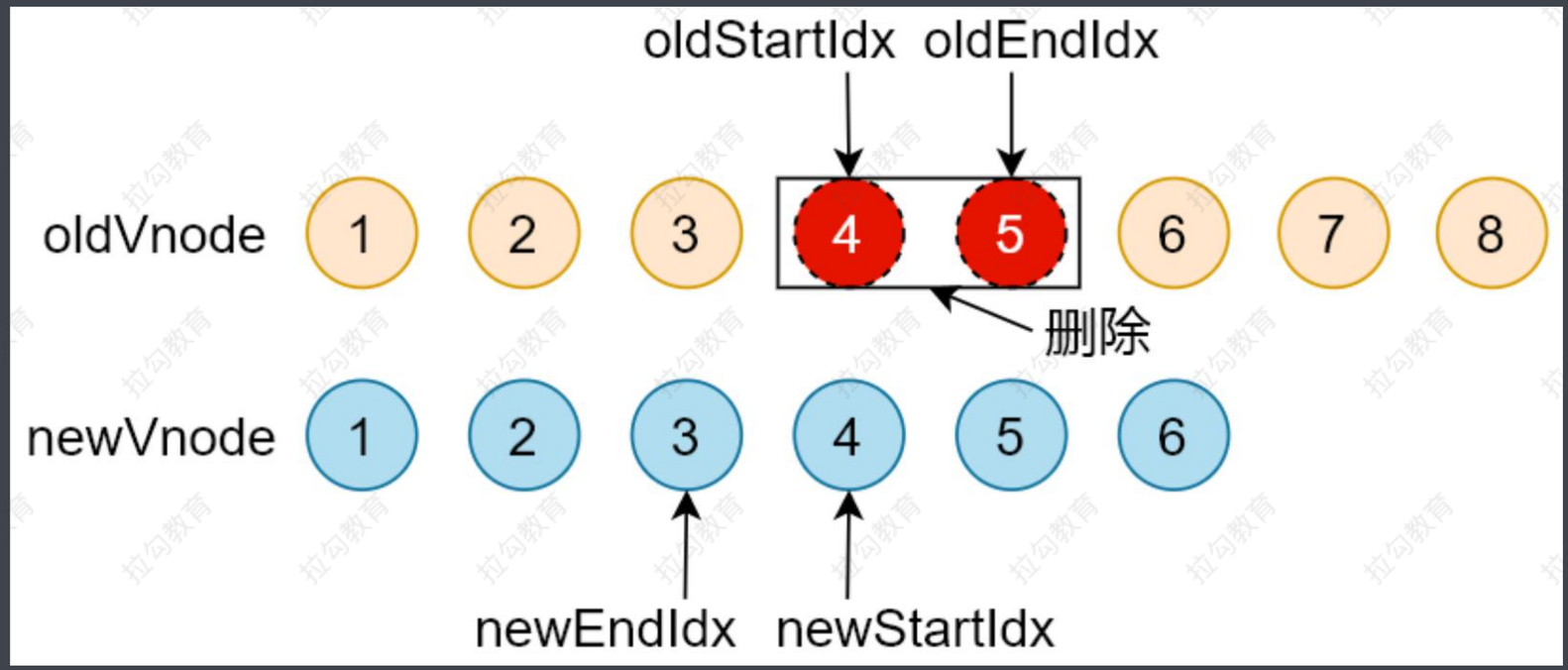
oldStartIdx > oldEndIdx

- 如果老节点的数组先遍历完(oldStartIdx > oldEndIdx)
  - 说明新节点有剩余，把剩余节点批量插入到右边



newStartIdx > newEndIdx

- 如果新节点的数组先遍历完(newStartIdx > newEndIdx)
  - 说明老节点有剩余，把剩余节点批量删除



Snabbdom 源码解析

updateChildren

Snabbdom 源码解析

调试 updateChildren

Snabbdom 源码解析

调试带 key 的情况

节点对比过程

Snabbdom 源码解析

Key 的意义