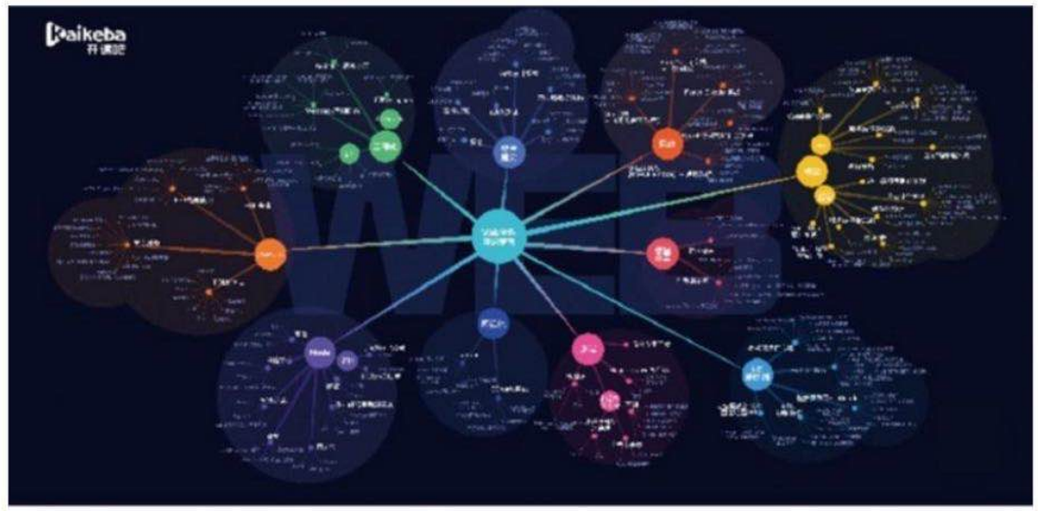
Vue训练营2

@大圣
8点正式开始



奖品: 独家定制web知识星球图谱鼠标垫
×5 份

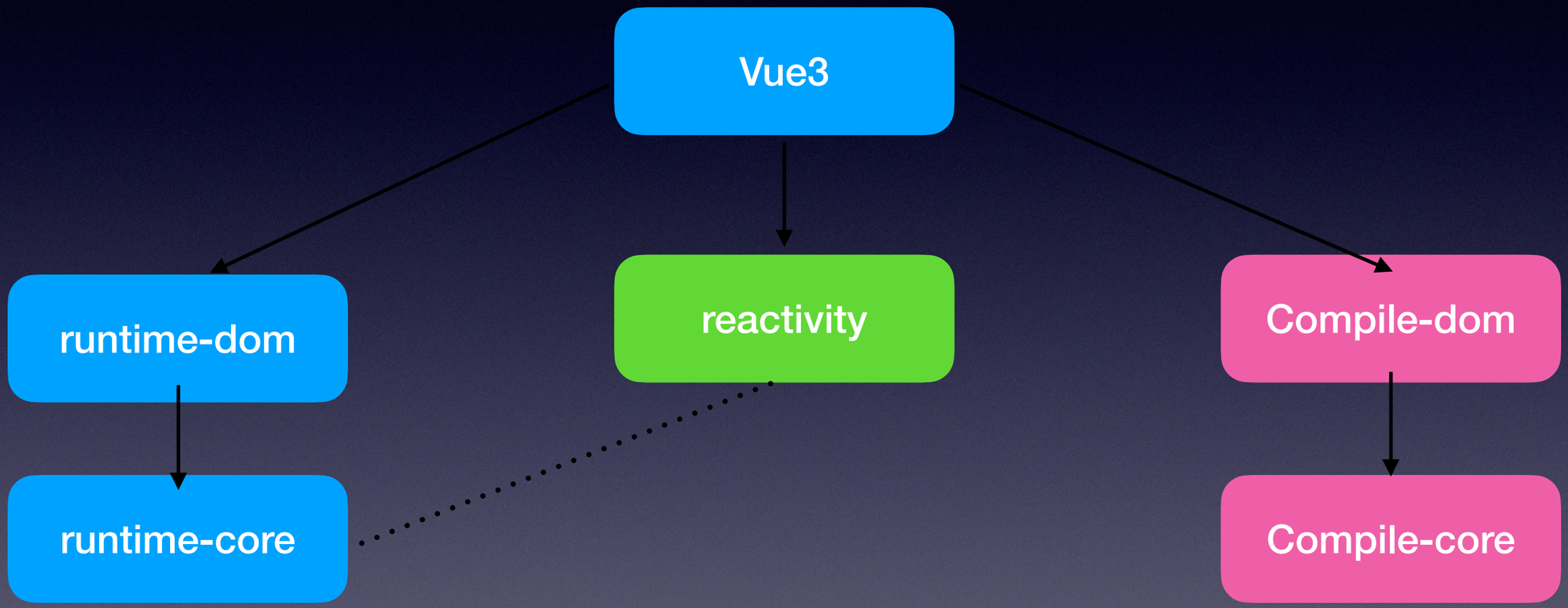
09月17日 22:31 自动开奖

长按识别小程序，参与抽奖

今日任务

- Vite 下一代脚手架工具
- 性能
- compiler
- 静态标记
- ssr对比
- 贡献代码心得

Vue3模块


Vue3
runtime-dom
reactivity
Compile-dom
runtime-core
Compile-core

尤小右

2018-4-18 21:27 来自 微博国际版

收藏 | 42 | 40 | 154

Compiler

- compiler目的只有一个，我们写的template 实际上是js，返回vdom
- 追本溯源，为什么vue/react需要编译

倔强青铜

- jquery dom查找修改 没啥模板的概念
- $(xx).html(yy)

秩序白银

- Underscore bdtemplate 字符串模板
- <%= 标记变量
- <% 标记js语法

```
<!-- <script type></script> -->
<script src="underscore.js"></script>
<script type="text/template" id="tpl">
  <ul class="list">
    <% _.each(obj, function(e, i, a){ %>
      <% if (i === 0) %>
        <li><%- e.name %>
      <% else if (i === a.length - 1) %>
        <li class="last-item"><%= e.name %></li>
      <% else %>
        <li><%= e.name %></li>
    <% }) %>
  </ul>
</script>

<script>
  var data = [{name: "开课吧大圣"}, {name: "最强王者"}];

  var compiled = _.template(document.getElementById("tpl").innerHTML);
  var html = compiled(data);
  // console.log(html)
  document.querySelector("div").innerHTML = html;
```

秩序白银

- 有数据变化，全量重新渲染即可
- 原理就是基本的字符串编译，拼接成js函数，然后new Function即可

```
  <script>
var tpl = `<% for(var i = 0; i < this.posts.length; i++) {
        var post = this.posts[i]; %>
        <% if(!post.expert){ %>
            <p>post is null</p>
        <% } else { %>
            <p><a href="#"><% post.expert %> at <% post.time %></a></p>
        <% } %>
    <% } %>`
```

```
function anonymous() {
  var r = [];
  for (var i = 0; i < this.posts.length; i++) {
    var post = this.posts[i];
    r.push("        ");
    if (!post.expert) {
      r.push("            <p>post is null</p>        ");
    } else {
      r.push("            <p><a href=\"#\">");
      r.push(post.expert);
      r.push(" at ");
      r.push(post.time);
      r.push("</a></p>        ");
    }
    r.push("    ");
  }
  return r.join("");
}
</script>
```

荣耀王者黄金

- 每次重新render成本太高
- dom是万恶之源
- 按需更新

```html
<body>

  <div id="app">
    开课吧
  </div>

  <script>
    let ele = document.getElementById('app')
    let arr = []
    for(let k in ele){
      arr.push(k)
    }
    console.dir(arr.length,arr.join(' '))
    // 一个简单的dom节点，竟然又200+属性 ，怪不的慢
  </script>
</body>
```

```js
// 其实用简单的对象就可以描述
let vdom = {
  type:"div",
  props:{ id:'app'},
  children:['开课吧']
}
```

top Filter Default levels

align title lang translate dir dataset hidden tabIndex accessKey draggable spellcheck autocapitalize contentEditable isCon dom.html:20
tentEditable inputMode offsetParent offsetTop offsetLeft offsetWidth offsetHeight style innerText outerText oncopy oncut onpaste onabo
rt onblur oncancel oncanplay oncanplaythrough onchange onclick onclose oncontextmenu oncuechange ondblclick ondrag ondragend ondragent
er ondragleave ondragover ondragstart ondrop ondurationchange onemptied onended onerror onfocus oninput oninvalid onkeydown onkeypress
onkeyup onload onloadeddata onloadedmetadata onloadstart onmousedown onmouseenter onmouseleave onmousemove onmouseout onmouseover onmo
useup onmousewheel onpause onplay onplaying onprogress onratechange onreset onresize onscroll onseeked onseeking onselect onstalled on
submit onsuspend ontimeupdate ontoggle onvolumechange onwaiting onwheel onauxclick ongotpointercapture onlostpointercapture onpointerd
own onpointermove onpointerup onpointercancel onpointerover onpointerout onpointerenter onpointerleave onselectstart onselectionchange
nonce click focus blur namespaceURI prefix localName tagName id className classList slot attributes shadowRoot assignedSlot innerHTML
outerHTML scrollTop scrollLeft scrollWidth scrollHeight clientTop clientLeft clientWidth clientHeight attributeStyleMap onbeforecopy o
nbeforecut onbeforepaste onsearch previousElementSibling nextElementSibling children firstElementChild lastElementChild childElementCo
unt onfullscreenchange onfullscreenerror onwebkitfullscreenchange onwebkitfullscreenerror setPointerCapture releasePointerCapture hasP
ointerCapture hasAttributes getAttributeNames getAttribute getAttributeNS setAttribute setAttributeNS removeAttribute removeAttributeN
S hasAttribute hasAttributeNS toggleAttribute getAttributeNode getAttributeNodeNS setAttributeNode setAttributeNodeNS removeAttributeN
ode closest matches webkitMatchesSelector attachShadow getElementsByTagName getElementsByTagNameNS getElementsByClassName insertAdjace
ntElement insertAdjacentText insertAdjacentHTML requestPointerLock getClientRects getBoundingClientRect scrollIntoView scroll scrollTo
scrollBy scrollIntoViewIfNeeded animate computedStyleMap before after replaceWith remove prepend append querySelector querySelectorAll
requestFullscreen webkitRequestFullScreen webkitRequestFullscreen part createShadowRoot getDestinationInsertionPoints ELEMENT_NODE ATT
RIBUTE_NODE TEXT_NODE CDATA_SECTION_NODE ENTITY_REFERENCE_NODE ENTITY_NODE PROCESSING_INSTRUCTION_NODE COMMENT_NODE DOCUMENT_NODE DOCU
MENT_TYPE_NODE DOCUMENT_FRAGMENT_NODE NOTATION_NODE DOCUMENT_POSITION_DISCONNECTED DOCUMENT_POSITION_PRECEDING DOCUMENT_POSITION_FOLLO
WING DOCUMENT_POSITION_CONTAINS DOCUMENT_POSITION_CONTAINED_BY DOCUMENT_POSITION_IMPLEMENTATION_SPECIFIC nodeType nodeName baseURI isC
onnected ownerDocument parentNode parentElement childNodes firstChild lastChild previousSibling nextSibling nodeValue textContent hasC
hildNodes getRootNode normalize cloneNode isEqualNode isSameNode compareDocumentPosition contains lookupPrefix lookupNamespaceURI isDe
faultNamespace insertBefore appendChild replaceChild removeChild addEventListener removeEventListener dispatchEvent

>

精细化分析

- 使用js对象描述标签
- 数据修改，不再全量覆盖，而是js先进行一次dom diff，然后算出需要修改最小的部分
- 相当于在template和浏览器之间，加了一层缓存

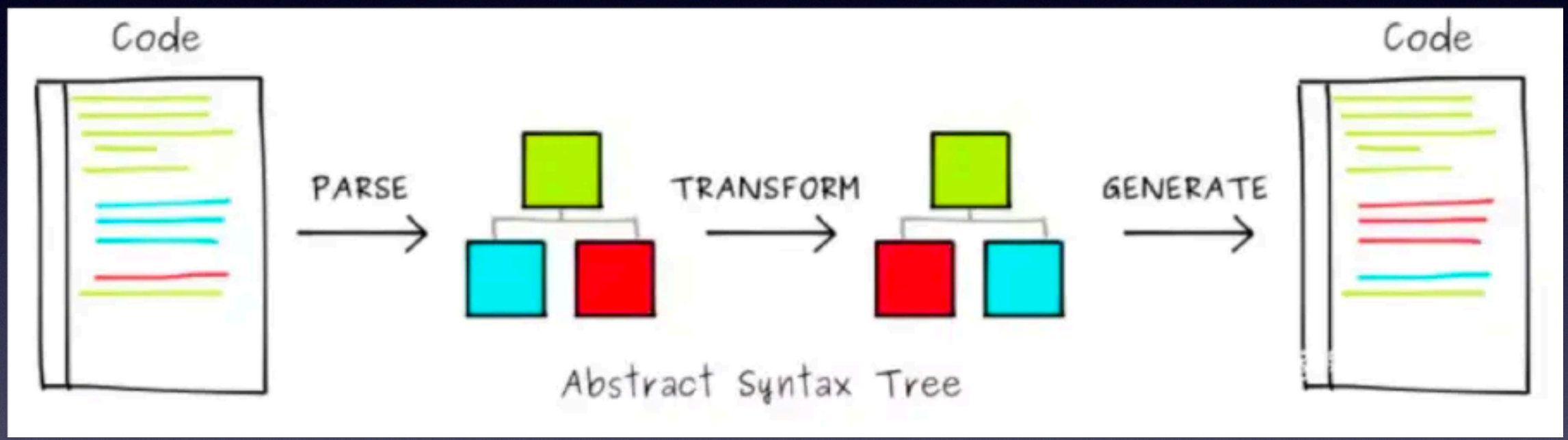
```
// 创建虚拟DOM
function createElement(type, props, children){
  return { type, props, children }
}
function render(dom) {
  let el = document.createElement(dom.type)
  for (let key in dom.props) {
      el.setAttribute(key, dom.props[key])
  }
  dom.children.forEach(child => {
      child = (typeof child=='object') ? render(child) : document.createTextNode(child)
      el.appendChild(child)
  })
  return el
}
let vdom = createElement('ul',{class:'list'},[
  createElement('li', {class:'item'},['item1']),
  createElement('li', {class:'item'},['item2']),
  createElement('li', {class:'item'},['item3'])
])
var el = render(vdom)
document.body.appendChild(el)
```

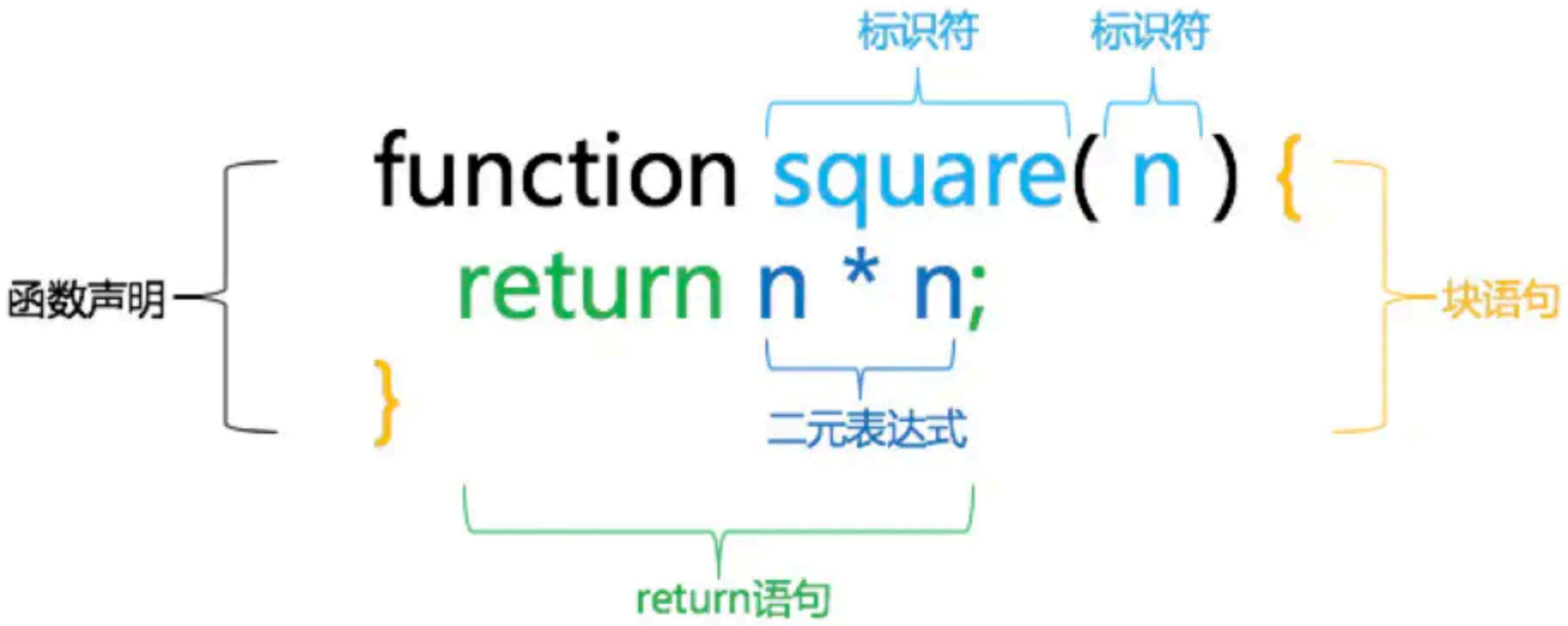
问题

- 用createElement创建元素 写起来太痛苦了，所以template=> vdom机制，应运而生
- https://template-explorer.vuejs.org/
- Vue2解析结果

编译

- 写的依然是html， 实际执行的是javascript
- template=> render函数，经典编译原理
- ast=> transform=>generate


Code
PARSE
TRANSFORM
GENERATE
Code
Abstract Syntax Tree




```
<div >
  <input v-model="name" />
  <h1>天气不错</h1>
  <p @click="handleChild"> {{name}} </p>
</div>

</template>
```

开始
自闭和
静态标签
开始
结束
事件
动态文本
结束

```
  const { code } = compile(template, {
    hoistStatic: true,
    onError(err: CompilerError) {
      if (__DEV__) {
        const message = `Template compilation error: ${err.message}`
        const codeFrame =
          err.loc &&
          generateCodeFrame(
            template as string,
            err.loc.start.offset,
            err.loc.end.offset
          )
        warn(codeFrame ? `${message}\n${codeFrame}` : message)
      } else {
        throw err
      }
    },
    ...options
  })

  // The wildcard import results in a huge object with every export
  // with keys that cannot be mangled, and can be quite heavy size-wise.
  // In the global build we know `Vue` is available globally so we can avoid
  // the wildcard object.
  const render = (__GLOBAL__
    ? new Function(code)()
    : new Function('Vue', code)(runtimeDom)) as RenderFunction
  return (compileCache[key] = render)
}
```

Compile

baseCompile

baseParse

v-model

V-on

V-for

静态提升

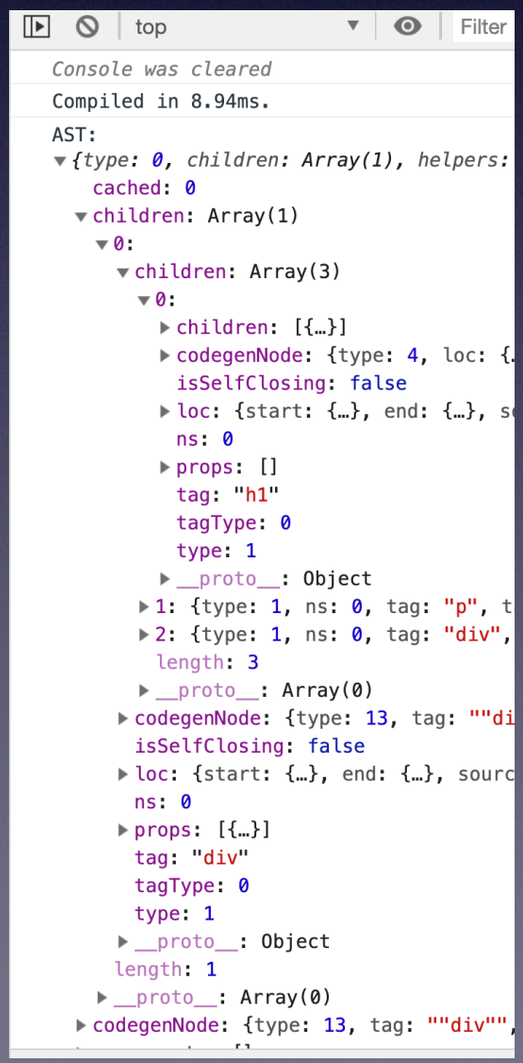
Template

AST

Transform

Generate

```
<div id="app">
    <h1>技术摸鱼</h1>
    <p>今天天气真不错</p>
    <div :id="userid">{{name}}</div>
</div>
```




```
import { createVNode as _createVNode, toDisplayString as
_toDisplayString, openBlock as _openBlock, createBlock as
_createBlock } from "vue"

const _hoisted_1 = { id: "app" }
const _hoisted_2 = /*#__PURE__*/_createVNode("h1", null, "技术摸鱼",
-1 /* HOISTED */)
const _hoisted_3 = /*#__PURE__*/_createVNode("p", null, "今天天气真不错
", -1 /* HOISTED */)

export function render(_ctx, _cache) {
  return (_openBlock(), _createBlock("div", _hoisted_1, [
    _hoisted_2,
    _hoisted_3,
    _createVNode("div", { id: _ctx.userid }, _toDisplayString(_ctx.
name), 9 /* TEXT, PROPS */, ["id"])
  ]))
}

// Check the console for the AST
```

```
const ast = isString(template) ? baseParse(template, options) : template
const [nodeTransforms, directiveTransforms] = getBaseTransformPreset(
  prefixIdentifiers
)
transform(ast, {
  ...options,
  prefixIdentifiers,
  nodeTransforms: [
    ...nodeTransforms,
    ...(options.nodeTransforms || []) // user transforms
  ],
  directiveTransforms: {
    ...directiveTransforms,
    ...(options.directiveTransforms || {}) // user transforms
  }
})

return generate(ast, {
  ...options,
  prefixIdentifiers
})
}
```

parse

baseParser

parseChildren

动态文本

截止标签

开始标签

普通文本

标签属性

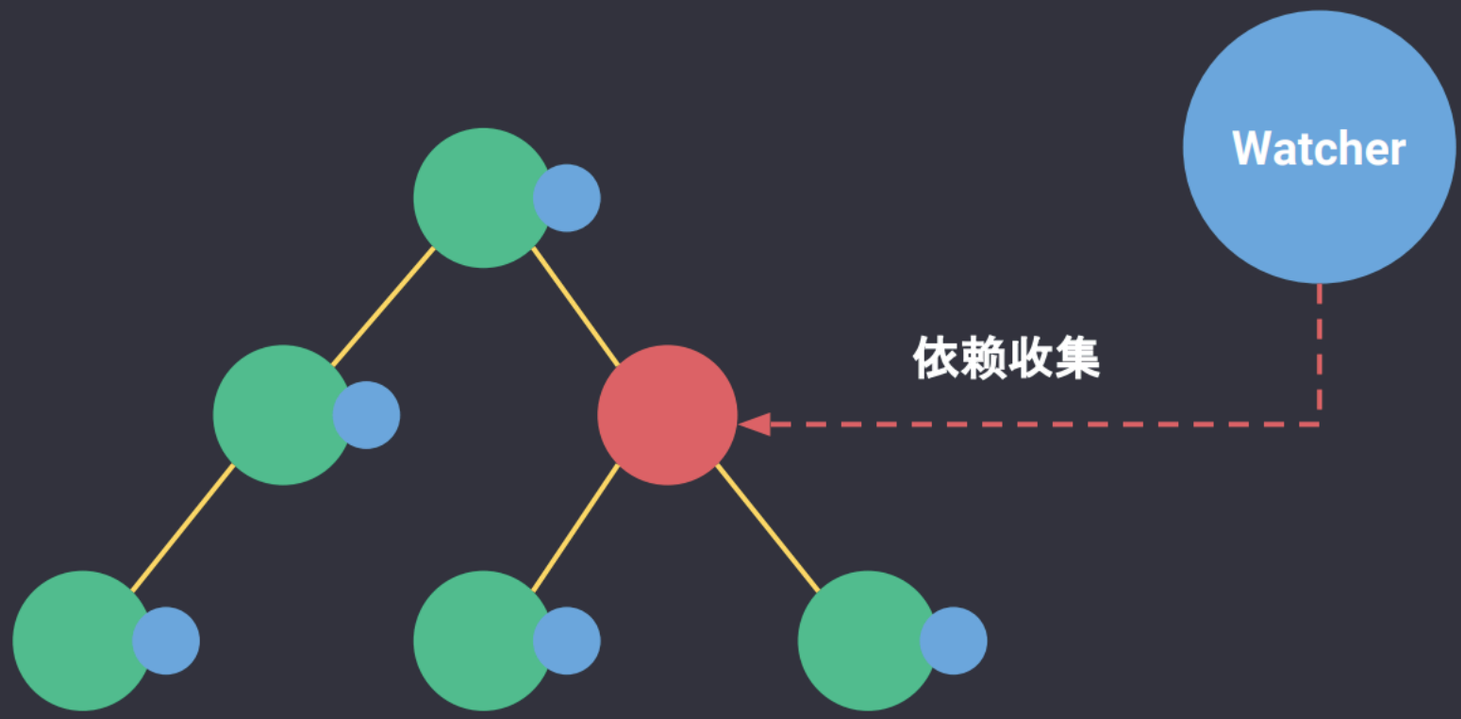
```
      node = parseInterpolation(context, mode)
    } else if (mode === TextModes.DATA && s[0] === '<') {
      // https://html.spec.whatwg.org/multipage/parsing.html#tag-open-state
      if (s.length === 1) {
        emitError(context, ErrorCodes.EOF_BEFORE_TAG_NAME, 1)
      } else if (s[1] === '!') {
        // https://html.spec.whatwg.org/multipage/parsing.html#markup-declaration-open-state
        if (startsWith(s, '<!--')) {
          node = parseComment(context)
        } else if (startsWith(s, '<!DOCTYPE')) {
          // Ignore DOCTYPE by a limitation.
          node = parseBogusComment(context)
        } else if (startsWith(s, '<![CDATA[')) {
          if (ns !== Namespaces.HTML) {
            node = parseCDATA(context, ancestors)
          } else {
            emitError(context, ErrorCodes.CDATA_IN_HTML_CONTENT)
            node = parseBogusComment(context)
          }
        } else {
          emitError(context, ErrorCodes.INCORRECTLY_OPENED_COMMENT)
          node = parseBogusComment(context)
        }
      } else if (s[1] === '/') {
        // https://html.spec.whatwg.org/multipage/parsing.html#end-tag-open-state
        if (s.length === 2) {
          emitError(context, ErrorCodes.EOF_BEFORE_TAG_NAME, 2)
        } else if (s[2] === '>') {
          emitError(context, ErrorCodes.MISSING_END_TAG_NAME, 2)
          advanceBy(context, 3)
          continue
        } else if (/[a-z]/i.test(s[2])) {
          emitError(context, ErrorCodes.X_INVALID_END_TAG)
          parseTag(context, TagType.End, parent)
          continue
        } else {
          emitError(
            context,
            ErrorCodes.INVALID_FIRST_CHARACTER_OF_TAG_NAME,
            2
          )
          node = parseBogusComment(context)
        }
      } else if (/[a-z]/i.test(s[1])) {
        node = parseElement(context, ancestors)
      } else if (s[1] === '?') {
        emitError(
          context,
          ErrorCodes.UNEXPECTED_QUESTION_MARK_INSTEAD_OF_TAG_NAME,
          1
```

Vdom diff

- Dom diff 目的是算出最小的修改
- 举个栗子

Vue3 vdom

传统 vdom 的性能瓶颈



- 虽然 Vue 能够保证触发更新的组件最小化，但在单个组件内部依然需要遍历该组件的整个 vdom 树

传统 vdom 的性能瓶颈

```
<template>
  <div id="content">
    <p class="text">Lorem ipsum</p>
    <p class="text">Lorem ipsum</p>
    <p class="text">{{ message }}</p>
    <p class="text">Lorem ipsum</p>
    <p class="text">Lorem ipsum</p>
  </div>
</template>
```

- Diff
  - Diff props of
  - Diff children of
    - Diff <p>
      - Diff props of <p>
      - Diff children of <p>
    - Repeat n times...

- 传统 vdom 的性能跟模版大小正相关，跟动态节点的数量无关。在一些组件整个模版内只有少量动态节点的情况下，这些遍历都是性能的浪费

传统 vdom 的性能瓶颈

```
function render() {
  const children = []
  for (let i = 0; i < 5; i++) {
    children.push(h('p', {
      class: 'text'
    }, i === 2 ? this.message : 'Lorum ipsum'))
  }
  return h('div', { id: 'content' }, children)
}
```

- 根本原因：JSX 和手写的 render function 是完全动态的，过度的灵活性导致运行时可以用于优化的信息不足

最简单的情况

```vue
<template>
  <div id="content">
    <p class="text">Lorem ipsum</p>
    <p class="text">Lorem ipsum</p>
    <p class="text">{{ message }}</p>
    <p class="text">Lorem ipsum</p>
    <p class="text">Lorem ipsum</p>
  </div>
</template>
```

- Diff <p> textContent

- 节点结构完全不会改变
- 只有一个动态节点

节点结构变化：v-if

```
<template>
 <div>
   <p class="text">Lorem ipsum</p>
   <p v-if="ok">
     <span>Lorem ipsum</span>
     <span>{{ message }}</span>
   </p>
 </div>
</template>
```

- Check <p v-if=”ok”>
  - Diff textContent

- v-if 外部：只有 v-if 是动态节点
- v-if 内部：只有 {{ message }} 是动态节点

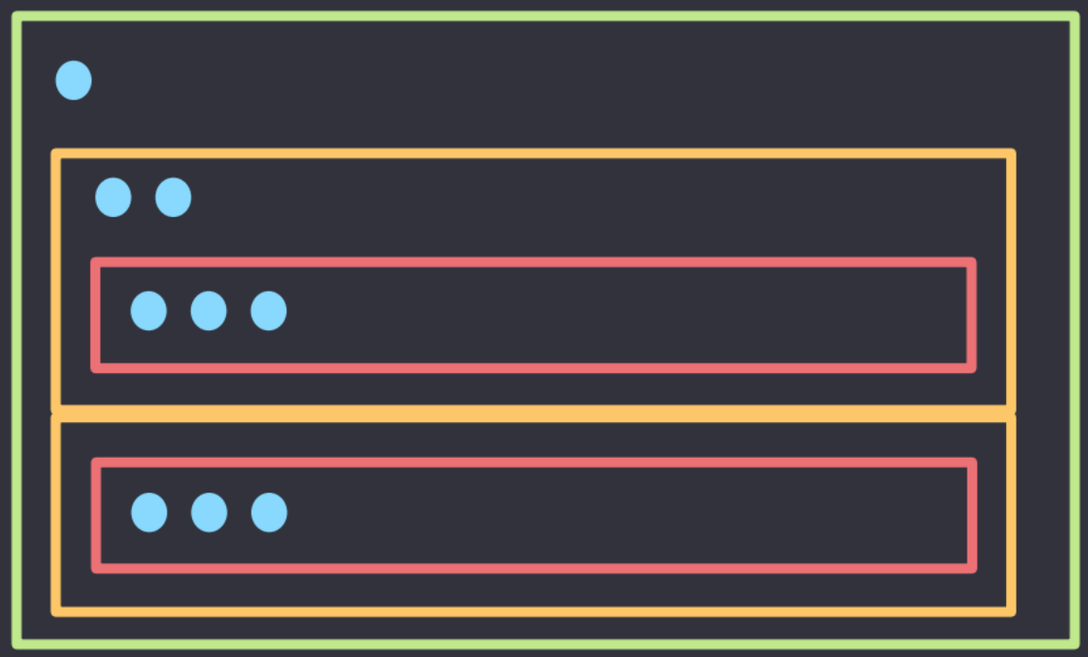
节点结构变化：v-for

```
<template>
 <div>
   <p class="text">Lorem ipsum</p>
   <p v-for="item of list">
     <span>Lorem ipsum</span>
     <span>{{ item.message }}</span>
   </p>
 </div>
</template>
```

- Diff <p v-for> children
  - Diff textContent
  - Repeat n times...

- v-for 外部：只有 v-for 是动态节点 (fragment)
- 每个 v-for 循环内部：只有 {{ item.message }} 是动态节点

Block tree “区块树”



- 将模版基于动态节点指令切割为嵌套的区块
- 每个区块内部的节点结构是固定的
- 每个区块只需要以一个 Array 追踪自身包含的动态节点

Before

```
<template>
  <div id="content">
    <p class="text">Lorem ipsum</p>
    <p class="text">Lorem ipsum</p>
    <p class="text">{{ message }}</p>
    <p class="text">Lorem ipsum</p>
    <p class="text">Lorem ipsum</p>
  </div>
</template>
```

- Diff
  - Diff props of
  - Diff children of
    - Diff <p>
      - Diff props of <p>
      - Diff children of <p>
    - Repeat n times...

After

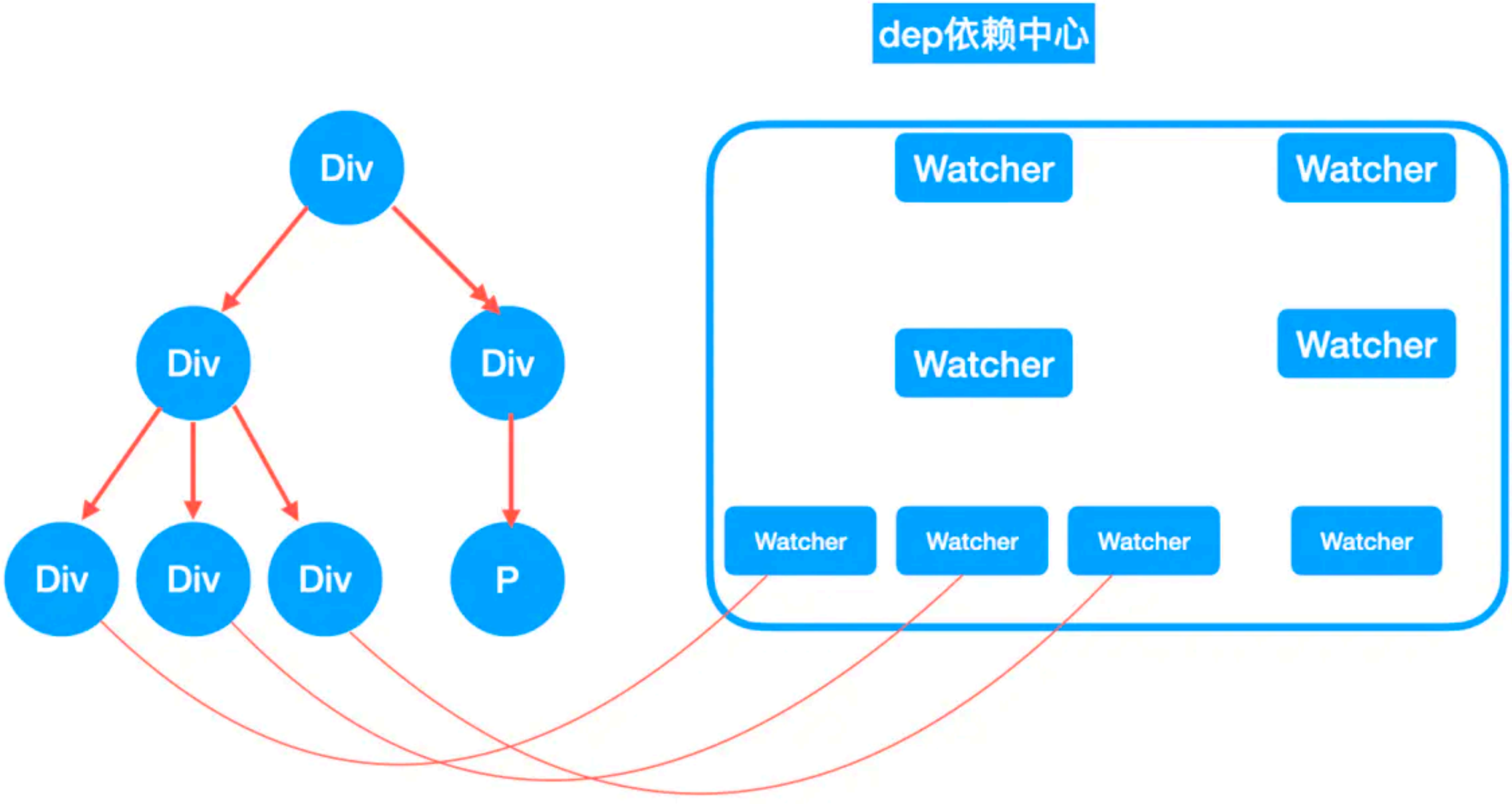
```
<template>
  <div id="content">
    <p class="text">Lorem ipsum</p>
    <p class="text">Lorem ipsum</p>
    <p class="text">{{ message }}</p>
    <p class="text">Lorem ipsum</p>
    <p class="text">Lorem ipsum</p>
  </div>
</template>
```

- Diff <p> textContent

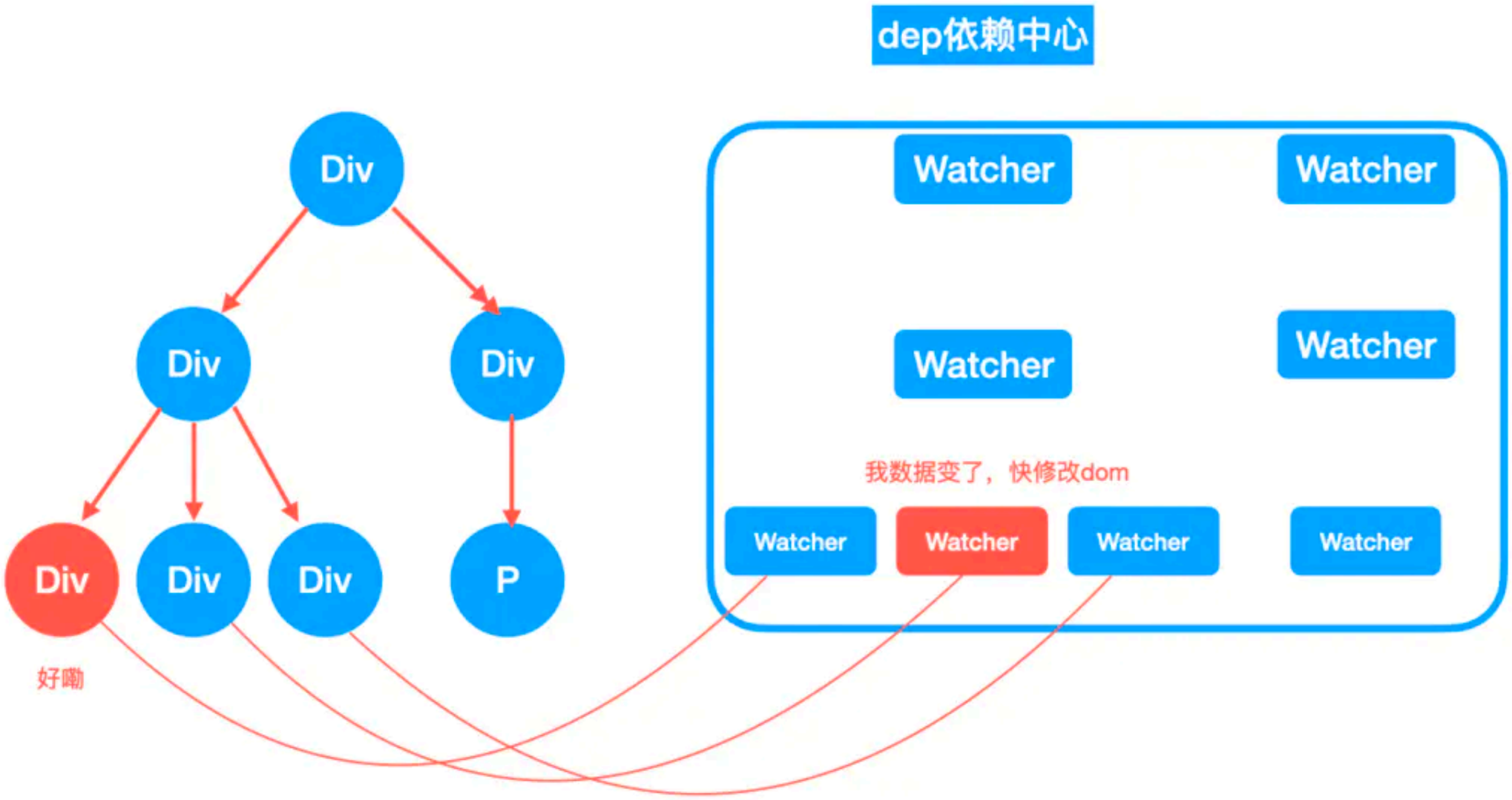
Vdom静态标记

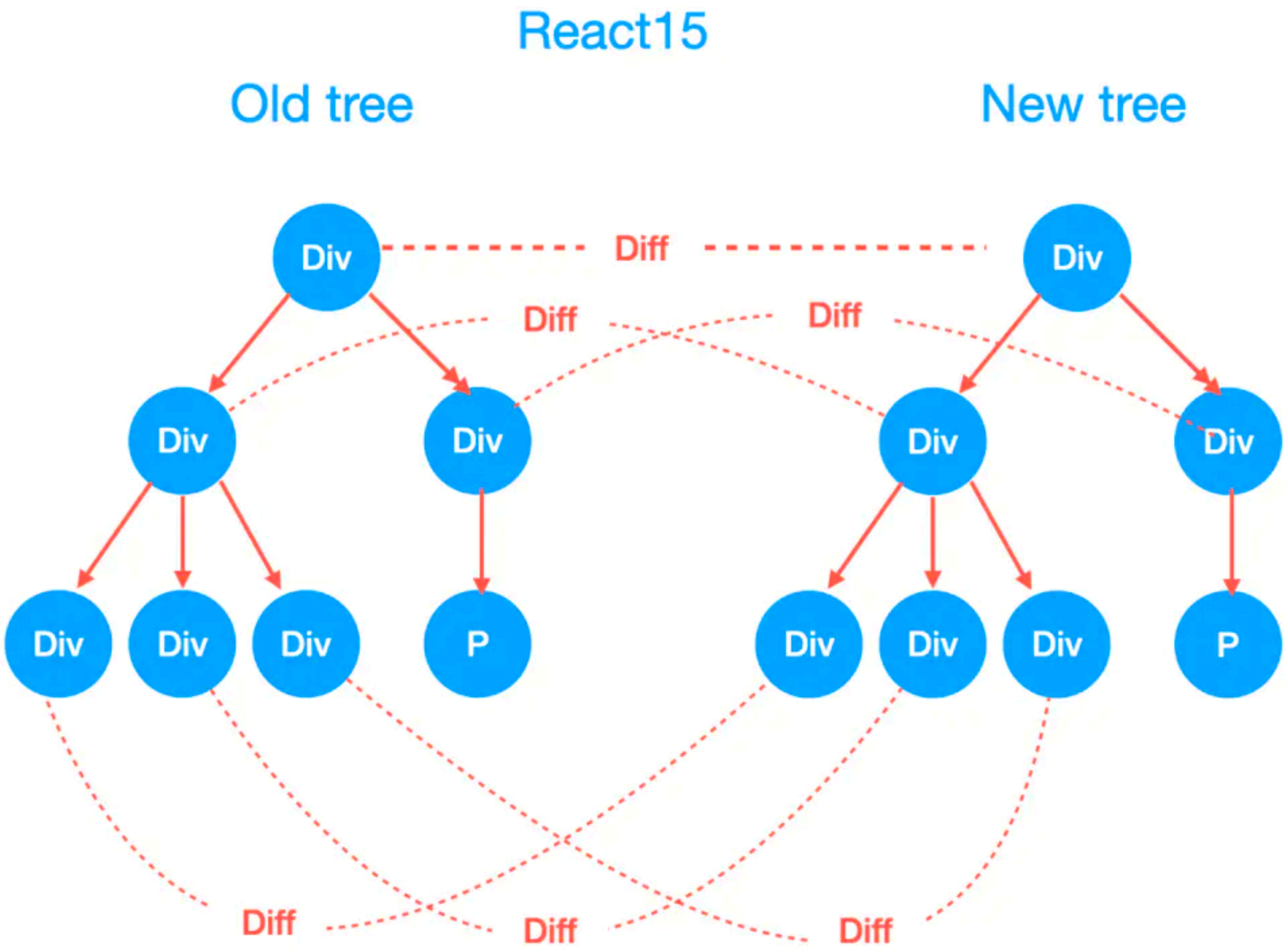
- 编译时优化 （灵感来源prepack.io）
- 静态标记
- vdom发展史 （Vue3为啥快）
- 在vue2静态标记的基础上，动态标签内部的静态标签，使用blcok来标记

Vue1.x


dep依赖中心
Div
Div
Div
Div
Div
Div
P
Watcher
Watcher
Watcher
Watcher
Watcher
Watcher
Watcher

Vue1.x


dep依赖中心
Watcher
Watcher
Watcher
Watcher
我数据变了，快修改dom
Watcher
Watcher
Watcher
Watcher
Div
Div
Div
Div
Div
Div
P
好嘞


React15
Old tree
New tree
Div
Diff
Diff
Diff
Div
Div
Div
Div
Div
Div
Div
Div
Div
Div
P
P
Diff
Diff
Diff

Vue2.x

Component

Component

Component

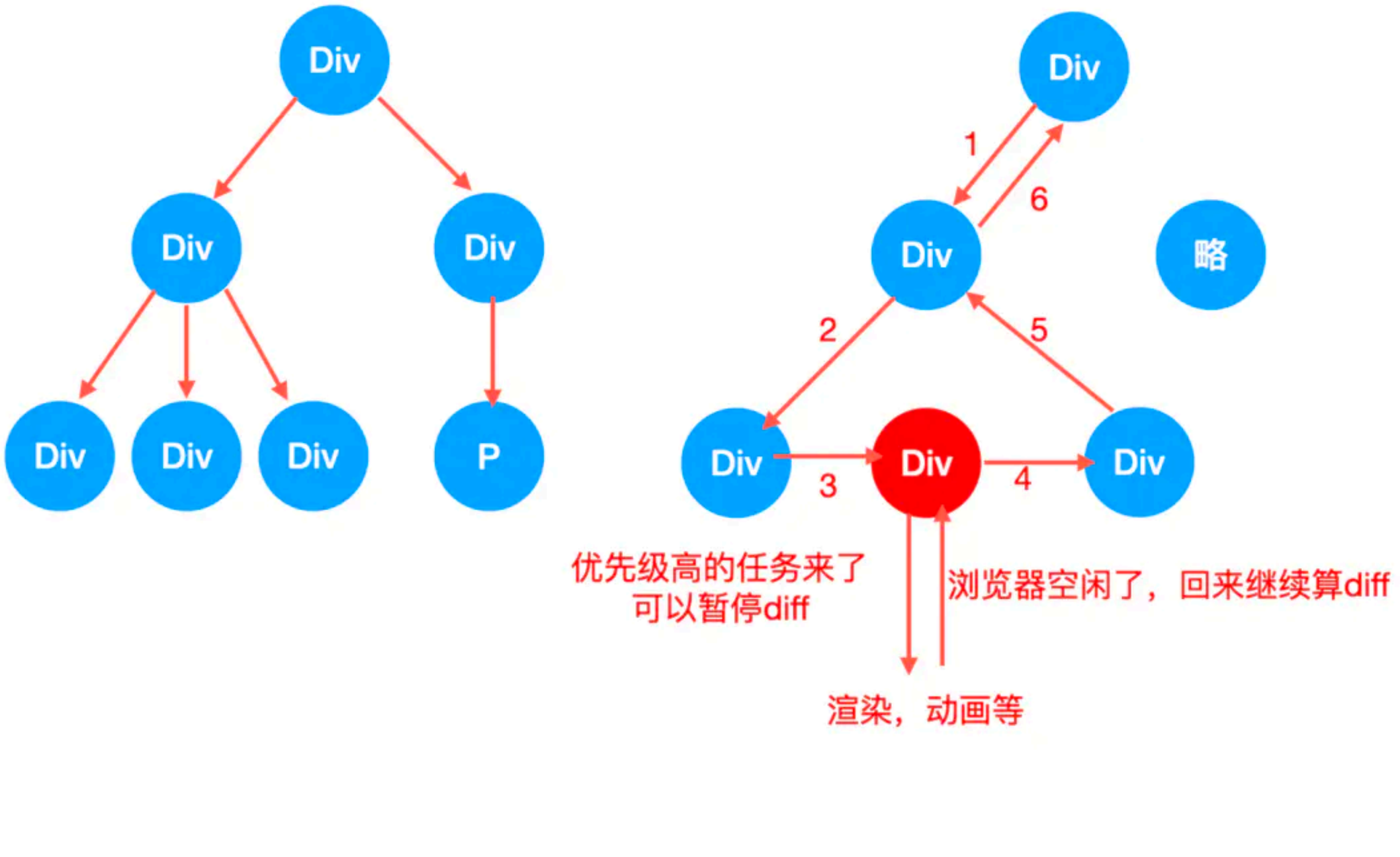
依赖中心

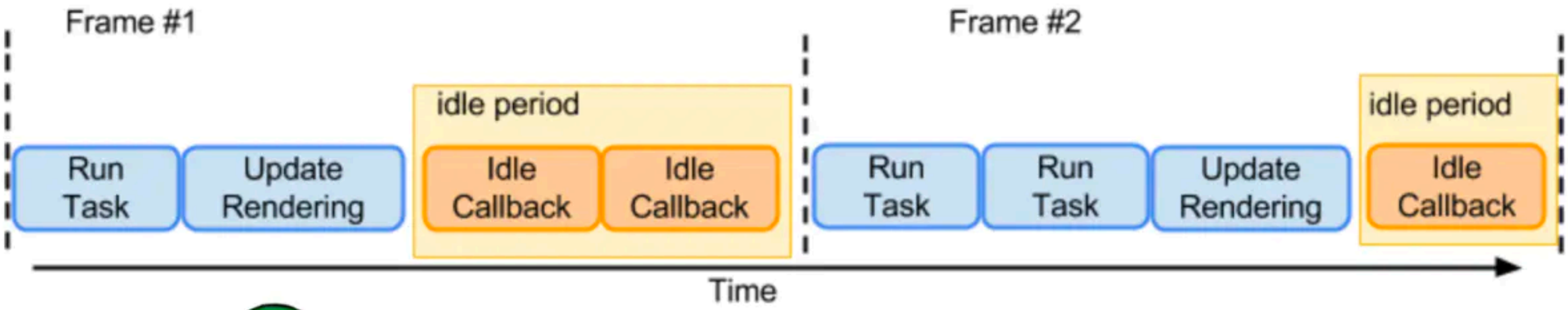
Watcher

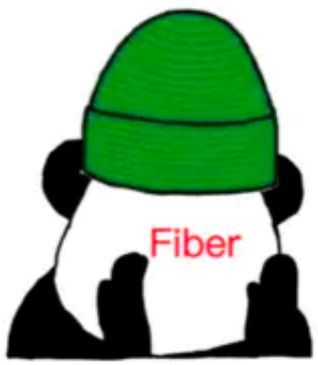
Watcher

Watcher

React Fiber


Old tree
Div
Div
Div
Div
Div
Div
P
Fiber tree
Div
1
6
Div
略
2
5
Div
Div
Div
3
4
优先级高的任务来了
可以暂停diff
浏览器空闲了，回来继续算diff
渲染，动画等



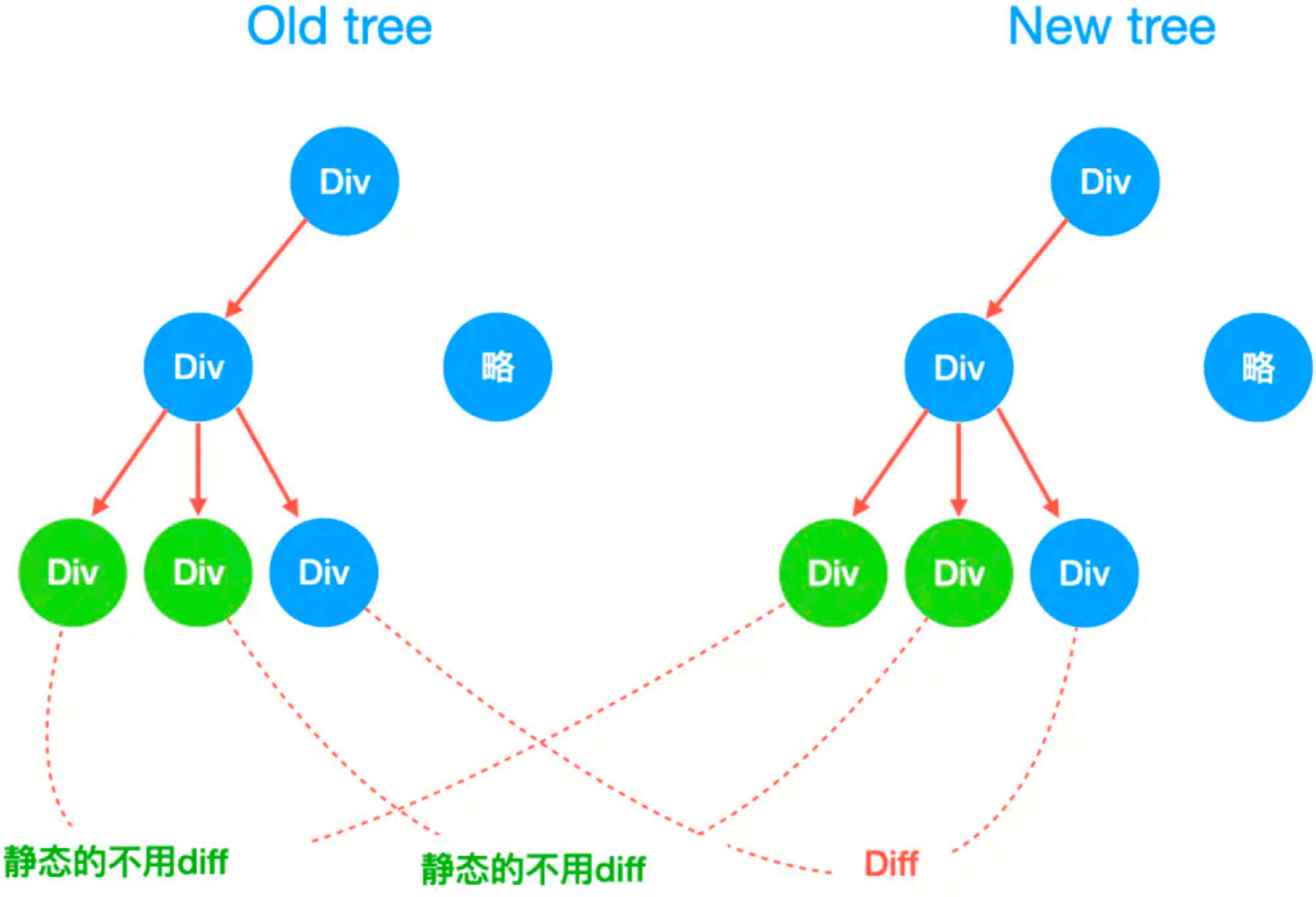


女神有空了
快计算diff

优先级高的富二代来了
比如动画，用户输入
把女神换回去

女神有空了
继续计算

Vue3.0


Old tree
New tree
Div
Div
略
Div
Div
Div
Div
Div
Div
Div
Div
略
Div
Div
Div
静态的不用diff
静态的不用diff
Diff

深夜话题

- 35岁危机
- 学习 VS 加班
- 成长经历
- 如何描述做的项目
- 早起看书
- vue3提源码

授课方式

直播

- 课程大纲
- 晚上8:30~10:30
- 直播互动答疑
- 上课听讲为主
- 100课时

训练营

- 独立知识 + 学员需求
- 针对某个话题的刻意训练
- 小群+强制交作业+奖励
- 录播
- 强制刷题 / 写作业
- 手写React ssr框架
- 算法刷题
- 浏览器原理
- 项目实战
- 网络协议
- 长期待续....

全栈架构师P6

新内容

- 微前端
- 前端基建
- 团队管理
- 软技能
- 终身学习

大厂思维

- Web安全实战
- 算法和数据结构
- 前端架构师体系构建
- 前端自动化测试
- javascript设计模式
- 面试技巧

广度

- 小程序 + 云开发
- Flutter实战
- 微信公众号
- ReactNative实战
- Webpack工程化
- Docker部署全栈应用

深度

- Vuejs全家桶源码
- 手写Eggjs MVC框架
- Webpack原理
- React16全家桶源码
- 小程序云开发实现支付
- 手写常见框架源码

实战

- Vue全家桶企业级实战
- Nodejs项目实战
- Kkb-cli脚手架开发
- React全家桶企业级实战
- Mysql && Mongodb
- Typescript

kaikeba 开课吧

Vue

Vue组件化设计

手写Vuex

Vue源码整体剖析

手写element表单

手写Vue-router

响应式原理

手写element弹窗

Typescript

模板编译+虚拟dom

权限控制

自动化测试

双向绑定原理

动态路由

服务端渲染ssr

Vue源码规范

kaikeba 开课吧

奖学金培优班

学多少 返多少

双重保障奖金
返还无忧

开课吧官方承诺 + 奖学金培优协议

满足你——

学习 返现 提升 涨薪

只限100名

kaikeba 开课吧

大圣训练营 专属福利

仅限前30名

三重好礼

以下优惠可叠加使用，请联系课程顾问详询

1 立减1000元 | 奖学金培优班适用

2 价值2899元 赠 全方位进阶之后端专题课

3 

价值79元 赠 《React工程师修炼指南》

详情联系你的课程顾问，开启你的学习提升之旅

kaikeba 开课吧

WEB全栈架构师

教研团队

Winter

前手机淘宝前端负责人

阿里 P8 高级技术专家，JavaScript 专家，带领团队开发了阿里巴巴开源项目 Weex 移动前端开发框架。

大圣老师

前百度和360前端架构师

9年前端开发经验，精通Vue/React、源码架构、小程序、移动端和Nodejs整个前端技术栈，对前端萌新如何快速进阶有丰富的经验，业内少有既精通技术又精通授课的大咖。

杨老师

前高伟达高级软件工程师

9年前端开发经验，精通Angular、Vue、React、Nodejs、Express、Koa等主流全栈开发框架，精通RN、Weex等Hybrid App开发框架，精通MySQL、MongoDB等，对模块化、组件化、自动化开发有深入实践和理解。

夏老师

前711电子商务平台前端负责人

8 年前端开发经验，涉及的行业主要有电商、证券、金融，曾参与许多大型电商项目的开发和设计工作，比如 711 电子商务平台、野村证券交易平台瑞穗银行信用卡结算平台、Reach-Max 广告数据分析平台等。涉及前后端多种开发项目，具有丰富的全栈开发及架构设计经验。

韩老师

前阿里/淘宝资深前端开发

7 年前端开发经验，任职开发期间，对大型电商平台、办公平台、性能优化、数据可视化、移动应用、SEO 等。有丰富的实战经验，比如阿里巴巴国际平台负责 SEO 业务对接和性能优化，手机淘宝 H5 技术开发，双 11、双 12 技术支撑，3.8 生活节前端负责人。精通 Vue、React、Node、小程序、公众号等主流前端技术栈。

高老师

前京东资深前端开发

6年前端开发经验，曾就职京东、万科、用友，精通React/Vue,曾参与主导过很多大型移动端和PC端项目，如活动模板开发、金融支付、财务系统、房产管理系统等，有着极其丰富的大型项目实战经验。


华为总部
联想总部
网易总部
百度总部
腾讯北京总部
新浪/微博总部
Kaikeba 开课吧 总部
IBM
快手总部
西二旗
百度大厦
滴滴总部


后厂理工学院
Caikeba开课吧


Kaikeba
开课吧
学习强国
Kaikeba
开课吧
HUIKE


字里行间
前进并敢于冒险。——慧科主张
知行合一
团结协作
美好的团建，就是带着团队一起打胜仗。
——慧科主张

深夜话题

- 面试学习法
- 35岁危机
- 学习 VS 加班
- 成长经历
- 如何描述做的项目
- 明天六点

奖品: React工程师修炼指南 ×3 份

09月17日 22:31 自动开奖

长按识别小程序，参与抽奖