# Vue环境安装

脚手架工具vue-cli,vue-cli自带webpack。 借助webpack工具开发大型开发项目。

>npm install –global vue-cli

>vue init webpack todolist //创建一个标准的webpack vue项目

错误： 此处vue –V报错，vue不是内部或者外部命令

方法：

1. 控制面板卸载nodejs

2. C:\Users\XXX\AppData下的npm和npm-cache删除

3. 下载安装新的nodejs→继续下面命令

node –v

npm –v

npm install –global vue-cli

vue init webpack todolist

cd todolist

npm run dev

|  |
| --- |
| C:\Users\XXXX>npm install --global vue-cli  npm WARN deprecated vue-cli@2.9.6: This package has been deprecated in favour of @vue/cli  npm WARN deprecated coffee-script@1.12.7: CoffeeScript on NPM has moved to "coffeescript" (no hyphen)  C:\Users\XXXX\AppData\Roaming\npm\vue -> C:\Users\XXXX\AppData\Roaming\npm\node\_modules\vue-cli\bin\vue  C:\Users\XXXX\AppData\Roaming\npm\vue-list -> C:\Users\XXXX\AppData\Roaming\npm\node\_modules\vue-cli\bin\vue-list  C:\Users\XXXX\AppData\Roaming\npm\vue-init -> C:\Users\XXXX\AppData\Roaming\npm\node\_modules\vue-cli\bin\vue-init  + vue-cli@2.9.6  added 241 packages from 206 contributors in 194.741s  C:\Users\XXXX>vue -V  2.9.6  C:\Users\XXXX>d:  D:\>cd D:\skillmap\vue  D:\skillmap\vue>vue init webpack todolist  ? Project name todolist  ? Project description A Vue.js project  ? Author sunasitA <XXXX@YYYYY>  ? Vue build standalone  ? Install vue-router? Yes  ? Use ESLint to lint your code? Yes  ? Pick an ESLint preset Standard  ? Set up unit tests Yes  ? Pick a test runner jest  ? Setup e2e tests with Nightwatch? Yes  ? Should we run `npm install` for you after the project has been created? (recommended) npm  vue-cli · Generated "todolist".  # Installing project dependencies ...  # ========================  npm WARN deprecated extract-text-webpack-plugin@3.0.2: Deprecated. Please use https://github.com/webpack-contrib/mini-css-extract-plugin  npm WARN deprecated browserslist@2.11.3: Browserslist 2 could fail on reading Browserslist >3.0 config used in other tools.  npm WARN deprecated bfj-node4@5.3.1: Switch to the `bfj` package for fixes and new features!  npm WARN deprecated json3@3.3.2: Please use the native JSON object instead of JSON 3  npm WARN deprecated flatten@1.0.2: I wrote this module a very long time ago; you should use something else.  npm WARN deprecated browserslist@1.7.7: Browserslist 2 could fail on reading Browserslist >3.0 config used in other tools.  npm WARN deprecated circular-json@0.3.3: CircularJSON is in maintenance only, flatted is its successor.  npm WARN deprecated socks@1.1.10: If using 2.x branch, please upgrade to at least 2.1.6 to avoid a serious bug with socket data flow and an import issue introduced in 2.1.0  npm WARN deprecated left-pad@1.3.0: use String.prototype.padStart()  > chromedriver@2.46.0 install D:\skillmap\vue\todolist\node\_modules\chromedriver  > node install.js  Current existing ChromeDriver binary is unavailable, proceding with download and extraction.  Downloading from file: https://chromedriver.storage.googleapis.com/2.46/chromedriver\_win32.zip  Saving to file: C:\Users\SUNNAN~1.FNS\AppData\Local\Temp\2.46\chromedriver\chromedriver\_win32.zip  ChromeDriver installation failed Error with http(s) request: Error: getaddrinfo ENOTFOUND chromedriver.storage.googleapis.com  npm WARN rollback Rolling back node-pre-gyp@0.12.0 failed (this is probably harmless): EPERM: operation not permitted, rmdir 'D:\skillmap\vue\todolist\node\_modules\fsevents\node\_modules'  npm WARN ajv-keywords@2.1.1 requires a peer of ajv@^5.0.0 but none is installed. You must install peer dependencies yourself.  npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.9 (node\_modules\fsevents):  npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.9: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})  npm ERR! code ELIFECYCLE  npm ERR! errno 1  npm ERR! chromedriver@2.46.0 install: `node install.js`  npm ERR! Exit status 1  npm ERR!  npm ERR! Failed at the chromedriver@2.46.0 install script.  npm ERR! This is probably not a problem with npm. There is likely additional logging output above.  npm ERR! A complete log of this run can be found in:  npm ERR! C:\Users\XXXX\AppData\Roaming\npm-cache\\_logs\2019-09-10T01\_09\_22\_219Z-debug.log  Running eslint --fix to comply with chosen preset rules...  # ========================  > todolist@1.0.0 lint D:\skillmap\vue\todolist  > eslint --ext .js,.vue src test/unit test/e2e/specs "--fix"  # Project initialization finished!  # ========================  To get started:  cd todolist  npm run dev  Documentation can be found at https://vuejs-templates.github.io/webpack  D:\skillmap\vue>cd todolist  D:\skillmap\vue\todolist>npm run dev  > todolist@1.0.0 dev D:\skillmap\vue\todolist  > webpack-dev-server --inline --progress --config build/webpack.dev.conf.js  13% building modules 25/31 modules 6 active ...!D:\skillmap\vue\todolist\src\App.vue{ parser: "babylon" } is deprecated; we now treat it as { parser: "bab 95% emitting  DONE Compiled successfully in 7880ms 9:10:05  I Your application is running here: http://localhost:8080 |

看到<http://localhost:8080> 成功！！

# Vue基本使用

## 什么是挂载点

new Vue({

el: '#app', //vue实例与id=app的DOM元素绑定，因此id=app的DOM元素被称为挂载点

router,

components: { App },

template: '<App/>'

})

## 什么是单文件组件，好处

Vue组件即一个.vue文件，单文件组件，好处：因为所有内容都在一个文件，可复用性强。

一个.vue文件中包含了一个组件全部的内容（模板，逻辑，样式）

|  |
| --- |
| <template>  </template>  <script>  export default{  }  </script>  <style>  </style> |

## Vue指令

* 1. v-bind或简写为：
  2. v-on:click简写为 @click
  3. v-model和v-bind的区别

v-model是动态的data《—》html的双向数据绑定（只能用在input, textarea, select上）。可以理解为v-on和v-bind的结合体。

v-bind将数据绑定到任何属性上。

* 1. v-show和v-if的区别

|  |  |  |
| --- | --- | --- |
|  | v-if | v-show |
| 手段 | 动态的向DOM树内添加或者删除DOM元素 | 通过设置DOM元素的display样式属性控制显隐 |
| 编译过程 | 切换有一个局部编译/卸载的过程，切换过程中合适地销毁和重建内部的事件监听和子组件 | 简单的基于css切换 |
| 编译条件 | 惰性的，如果初始条件为假，则什么也不做；只有在条件第一次变为真时才开始局部编译 | 是在任何条件下（首次条件是否为真）都被编译，然后被缓存，而且DOM元素保留 |
| 性能消耗 | 有更高的切换消耗 | 有更高的初始渲染消耗 |
| 使用场景 | 适合运营条件不大可能改变 | 适合频繁切换 |

## 组件化开发

Step1：父组件引入子组件import todoItem from './components/todoItem';

Step2:　components: {

"todo-item": todoItem

},

以上代码对局部组件进行声明，使用todo-item标签的形式使用todoItem子组件

「注意点」

1. Component的data是函数，否则报错

[Vue warn]: The "data" option should be a function that returns a per-instance value in component definitions.

1. 父子组件数据传值

■父->子： 属性传值

1）<todo-item

v-for="(item, index) of list"

:content="item"

:index="index"

@delete="DeleteHandle"

>

2）子组件接收父组件的传值

export default {

props: ['content', 'index']

}

■子->父： 属性传值

1. this.$emit('delete', this.index);对外发布一个delete方法，并传参数index
2. 父组件订阅delete方法

<todo-item

v-for="(item, index) of list"

:content="item"

:index="index"

@delete="DeleteHandle"

>

➂兄弟组件间传值

父组件index.vue

子组件LeftTree.vue, RightMain.vue

试图从LeftTree.vue —》RightMain.vue传值，即兄弟组件间传值

This.$emit(‘passLeftData’,leftData)

LeftTree.vue

index.vue

props:[‘passLeftData’]

RightMain.vue

v-on:passLeftData

data:leftData

子组件LeftTree.vue

|  |
| --- |
| data: function(){      return {        colorVal: this.bcolor,        fontColor: this.value,        leftData: '我是left中的data值'      }    }  <button @click="cgColor">改变背景</button>   methods: {      cgColor: function(){        this.$emit('passLeftData', this.leftData);      }    } |

父组件 index.vue

|  |
| --- |
| <div class='index'>      <left-tree        v-on:passLeftData='getLeftData'      >        这是左侧导航      </left-tree>      <right-main        :leftDataFromParent = 'ld'      >      </right-main>    </div>  data: function(){      return {        bcolor: '#0787aa',        fcolor: 'white',        ld: ''      }    },    methods: {      getLeftData: function(leftData) {        this.ld = leftData;      }    }, |

子组件RightMain.vue

|  |
| --- |
| export default {    name: 'RightMain',    props: ['leftDataFromParent']  }   <div>{{leftDataFromParent}}</div> |

④使用v-model给子组件传值，子组件的props中只能用value接收

|  |
| --- |
| 父组件  <template>  <div class='index'>  <!—两种方式给子组件传值-->      <left-tree        :bcolor='bcolor'        v-model='fcolor'        v-on:passColor2Parent='colorChanged' <!—监听子组件方法，并用自定义方法colorChanged接收子组件传来的参数-->      >        这是左侧导航      </left-tree>      <right-main></right-main>    </div>  </template>  <script>  import LeftTree from '@/components/LeftTree'  import RightMain from '@/components/RightMain'  export default {    name: 'Index',    data: function(){      return {        bcolor: '#0787aa',        fcolor: 'white'      }    },    methods: {      colorChanged: function(bcolor, fcolor) {//参数        console.log(bcolor + ' ' +fcolor);      }    },    components: {      'left-tree': LeftTree,      'right-main': RightMain    }  }  </script>  <!-- Add "scoped" attribute to limit CSS to this component only -->  <style scoped>    \*{      margin-left: 6px;    }    .index {      display: flex;      flex-direction: row;      height: 98vh;      width: 98vw;    }  </style>  子组件  <template>    <div class='left' :style="{ backgroundColor: colorVal, color: fontColor}">      <slot></slot>      <button @click="cgColor">改变背景</button>    </div>  </template>  <script>  import {isPrimitive} from '../util.js'  // import isPrimitive from '../util.js' 方法的引入没有加{},报错Object(...) is not a function  at VueComponent.created  export default {    name: 'LeftTree',    props: ['bcolor', 'value'],//父组件使用v-model传递的参数用value存放    data: function(){      return {        colorVal: this.bcolor,        fontColor: this.value      }    },    methods: {      cgColor: function(){//不能直接改变父组件传的参数，但是可以使用data接收后改变        this.colorVal = (this.colorVal === 'lightgrey') ? '#0787aa' : 'lightgrey';        this.fontColor = (this.colorVal === 'lightgrey') ? 'black' : 'white';        this.$emit('passColor2Parent', this.colorVal, this.fontColor)      }    },    created (){        // alert( isPrimitive(11) );    }  }  </script>  <style scoped>    .left{      flex: 1;      border: 1px solid gray;      color: white;    }  </style> |

➄Sync实现父子间数据同步

Home.vue

<parent :age.sync="age1"></parent>

data: function() {

        return {

            age1: 31

    }

parentComponent.vue

<span>{{age}}</span>

export default {

    name: "parentComponent",

    props: ["age"],

    methods: {

        clickHandle(){

            this.$emit('update:age', 11)

        }

    }

}

多属性

Home.vue

<parent :parentData.sync="parentData"></parent>

parentData: {

                name: 'papa',

                age: 31

            },

parentComponent.vue

<span>{{parentData.name}} - {{parentData.age}}</span>

props: ["parentData"],

methods: {

        clickHandle(){

            this.$emit('update:parentData', {

                name: "tom",

                age: 11

            });

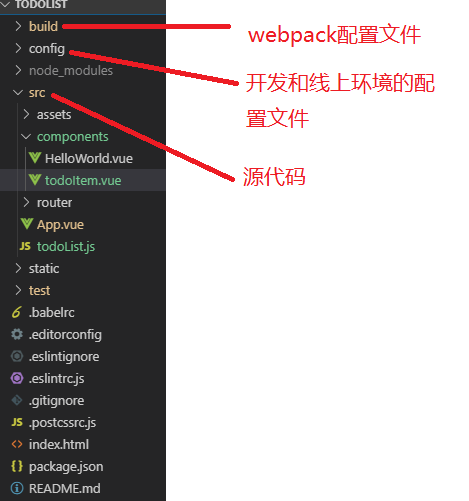
        }

    }

## Vue的Style写法

# 3. Vue源码解析

## 1. 源码目录



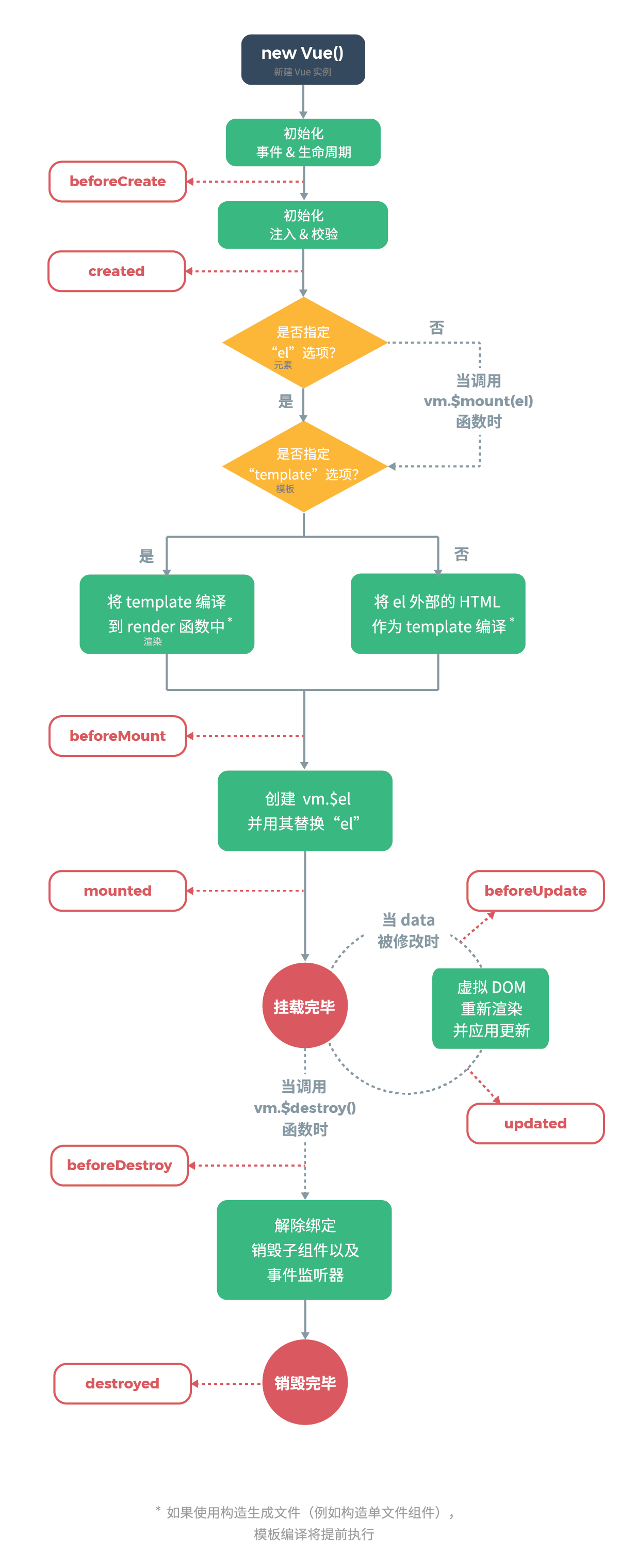
## initLifeCycle方法

用来初始化一些生命周期相关的属性，以及为parent,child等属性赋值。

|  |
| --- |
| **export function** *initLifecycle* (vm: Component) {  **const *options*** = **vm**.$options   *// locate first non-abstract parent* let ***parent*** = ***options***.parent  // 非抽象组件，且有父组件  **if** (***parent*** && !***options***.abstract) {  // 父组件是抽象组件且存在父组件，继续循环直到找到非抽象的parent，将其父组件付给parent  **while** (***parent***.$options.abstract && ***parent***.$parent) {  ***parent*** = ***parent***.$parent  }  ***parent***.$children.push(**vm**)// 将当前Vue实例存到第一个找到的非抽象组件中  }   **vm**.**$parent** = ***parent // 建立此实例和父实例之间的关系* vm**.**$root** = ***parent*** ? ***parent***.$root : **vm   vm**.**$children** = []  **vm**.**$refs** = {}   **vm**.**\_watcher** = **null  vm**.**\_inactive** = **null // 当前组件的keep-alive状态，与kepp-alive的值相反  vm**.**\_directInactive** = **false  vm**.**\_isMounted** = **false // 当前实例是否完成挂载  vm**.**\_isDestroyed** = **false // 当前实例是否被销毁  vm**.**\_isBeingDestroyed** = **false // 当前实例是否正在被销毁** } |

抽象组件，如keep-alive，自身不会渲染成一个DOM元素

## 生命周期图



数据劫持，属性和方法的运算，watch/event 事件回调OK，没开始挂载阶段

实例初始化OK

数据劫持,事件配置前

数据更新，patch前

虚拟 DOM重新渲染和打补丁后，组件 DOM 已更新

el被新创建的vm.$el 替换

挂载前

## initRender->\_createElement

直接看核心方法\_createElement，由于过长，分段解析。

|  |
| --- |
| **data是合法的值(不是未定义和null)，且data已经有\_\_ob\_\_属性即data已经被加到数据劫持中，报错**  **if** (isDef(**data**) && isDef((**data**: ***any***).\_\_ob\_\_)) {  process.env.NODE\_ENV !== **'production'** && warn(  **`Avoid using observed data object as vnode data:** ${JSON.stringify(data)}**\n`** +  **'Always create fresh vnode data objects in each render!'**,  **context** )  **return** createEmptyVNode()// 创建一个空节点（text空，isComment未true） }  // tag不存在，创建空节点  **if** (!**tag**) {  *// in case of component :is set to falsy value* **return** createEmptyVNode() } |

|  |
| --- |
| **if** (Array.isArray(**children**) &&  **typeof children**[0] === **'function'**) {  **data** = **data** || {}  **data**.**scopedSlots** = { **default**: **children**[0] }  **children**.**length** = 0 }  **if** (**normalizationType** === ***ALWAYS\_NORMALIZE***) {  **children** = normalizeChildren(**children**) } **else if** (**normalizationType** === ***SIMPLE\_NORMALIZE***) {  **children** = simpleNormalizeChildren(**children**) } |

|  |
| --- |
| **if** (**typeof tag** === **'string'**) {  let ***Ctor*** */\*获取tag的名字空间\*/* ***ns*** = config.**getTagNamespace**(**tag**)  */\*判断是否是保留的标签\*/* **if** (config.**isReservedTag**(**tag**)) {  *// platform built-in elements  /\*如果是保留的标签则创建一个相应节点\*/* ***vnode*** = **new** VNode(  config.**parsePlatformTagName**(**tag**), **data**, **children**,  **undefined**, **undefined**, **context** )  } **else if** (isDef(***Ctor*** = resolveAsset(**context**.$options, **'components'**, **tag**))) {  *// component  /\*从vm实例的option的components中寻找该tag，存在则就是一个组件，创建相应节点，Ctor为组件的构造类\*/* ***vnode*** = createComponent(***Ctor***, **data**, **context**, **children**, **tag**)  } **else** {  *// unknown or unlisted namespaced elements  // check at runtime because it may get assigned a namespace when its  // parent normalizes children  /\*未知的元素，在运行时检查，因为父组件可能在序列化子组件的时候分配一个名字空间\*/* ***vnode*** = **new** VNode(  **tag**, **data**, **children**,  **undefined**, **undefined**, **context** )  } } **else** {  *// direct component options / constructor  /\*tag不是字符串的时候则是组件的构造类\*/* ***vnode*** = createComponent(**tag**, **data**, **context**, **children**) } **if** (isDef(***vnode***)) {  */\*如果有名字空间，则递归所有子节点应用该名字空间\*/* **if** (***ns***) *applyNS*(***vnode***, ***ns***)  **return** vnode } **else** {  */\*如果vnode没有成功创建则创建空节点\*/* **return** createEmptyVNode() } |

vNode类

虚拟节点，拥有Dom元素各种属性的javascript对象。

空节点： text=”” isComment=true

文本节点： text有值

Tag在Vue实例的components属性值中，即tag作为子组件引入：createComponent创建组件节点

创建函数组件

createComponent

|  |
| --- |
| **//创建组件节点**  **export function** *createComponent* (Ctor,**data**,**context**,**children**,**tag** ) {  */\*没有传组件构造类直接返回\*/* **if** (isUndef(**Ctor**)) {  **return** }  */\* src/core/instance/init.js的下面代码段将Vue上的option合并到vm.$options  vm.$options = mergeOptions(  resolveConstructorOptions(vm.constructor),  options || {},  vm)  结合src/core/global-api/index.js Vue.options.\_base = Vue  将基类Vue给baseCtor \*/*  **const *baseCtor*** = **context**.**$options**.**\_base** *// plain options object: turn it into a constructor* **if** (isObject(**Ctor**)) {  //Vue.extend缓存sub的作用：防止多次执行 Vue.extend 的时候对同一个子组件重复构造  **Ctor** = ***baseCtor***.extend(**Ctor**)   }   *// if at this stage it's not a constructor or an async component factory,  // reject.  /\*如果在该阶段Ctor依然不是一个构造函数或者是一个异步组件工厂则直接返回\*/* **if** (**typeof Ctor** !== **'function'**) {  **if** (process.env.NODE\_ENV !== **'production'**) {  warn(**`Invalid Component definition:** ${String(Ctor)}**`**, **context**)  }  **return** }   *// async component  /\*处理异步组件\*/* **if** (isUndef(**Ctor**.**cid**)) {  **Ctor** = resolveAsyncComponent(**Ctor**, ***baseCtor***, **context**)  **if** (**Ctor** === **undefined**) {  *// return nothing if this is indeed an async component  // wait for the callback to trigger parent update.  /\*如果这是一个异步组件则不会返回任何东西（undefined），直接return掉，等待回调函数去触发父组件更新。s\*/* **return** }  }   *// resolve constructor options in case global mixins are applied after  // component constructor creation* resolveConstructorOptions(**Ctor**)   **data** = **data** || {}   *// transform component v-model data into props & events* **if** (isDef(**data**.*model*)) {  *transformModel*(**Ctor**.**options**, **data**)  }   *// extract props* **const *propsData*** = extractPropsFromVNodeData(**data**, **Ctor**, **tag**)   *// functional component* **if** (isTrue(**Ctor**.**options**.functional)) {  **return** createFunctionalComponent(Ctor, propsData, data, context, children)  }   *// extract listeners, since these needs to be treated as  // child component listeners instead of DOM listeners* **const *listeners*** = **data**.**on** *// replace with listeners with .native modifier* **data**.**on** = **data**.nativeOn   **if** (isTrue(**Ctor**.**options**.**abstract**)) {  *// abstract components do not keep anything  // other than props & listeners* **data** = {}  }   *// merge component management hooks onto the placeholder node  mergeHooks*(**data**)   *// return a placeholder vnode* **const *name*** = **Ctor**.**options**.**name** || **tag  const *vnode*** = **new** VNode(  **`vue-component-**${**Ctor**.**cid**}${***name*** ? **`-**${name}**`** : **''**}**`**,  **data**, **undefined**, **undefined**, **undefined**, **context**,  { Ctor, propsData, listeners, tag, children }  )  **return** vnode } |

## 路由

概要：子路由和传值等的介绍

路由的使用分为四部：

Step1：定义路由路径集合

var routes = [{path: ‘’, component:}, …]

Step2：新建一个路由对象

var routers = new VueRouter({routes})

Step3：将路由挂载到Vue实例上

new Vue({

el: ‘#app’,

routers

})

传值的方式有两种

方法1：

var routes = [{path: ‘/home/:name/:id’, component:}, …]

<router-link to=’/home/tom/123’>

<div>{{ this.$route.parames.name }} – {{ this.$route.parames.id }}</div>

方法2：

var routes = [{path: ‘/home’, component:}, …]

<router-link to=’/home?name=tom&id=123’>

<div>{{ this.$route.query.name }} – {{ this.$route.query.id }}</div>

完整的代码，这边是使用vue脚手架生成项目的代码

vue\_demo\src\router\index.js

|  |
| --- |
| import Vue from 'vue'  import Router from 'vue-router'  import Index from '@/Index'  import Home from './../components/routers/home.vue'  import User from './../components/routers/user.vue'  import userLogin from './../components/routers/userLogin.vue'  import userRegister from './../components/routers/userRegister.vue'  Vue.use(Router)  export default new Router({    routes: [      {        path: '/',        name: 'Index',        component: Index      },      {        path: '/home',        name: 'Home',        component: Home      },      {        path: '/user',        name: 'User',        component: User,        children: [          {            path: 'login/:name/:id',            name: 'userLogin',            component: userLogin          },{            path: 'register',            name: 'userRegister',            component: userRegister          }        ]      }    ]  }) |

vue\_demo\src\components\RightMain.vue

|  |
| --- |
| <template>    <div class='right'>      <router-link to="/home">首页</router-link>      <router-link to="/user">用户页面</router-link>      <div>        <router-view></router-view>      </div>    </div>  </template>  <script>  export default {    name: 'RightMain'  }  </script>  <!-- Add "scoped" attribute to limit CSS to this component only -->  <style scoped>    .right{      flex: 3;      border: 1px solid gray;      margin-left: 5px;    }  </style> |

vue\_demo\src\components\routers\home.vue

|  |
| --- |
| <template>      <div>首页</div>  </template>  <script>  export default {      name: 'home'  }  </script> |

vue\_demo\src\components\routers\user.vue

|  |
| --- |
| <template>      <div>          <router-link to='/user/login/tom/123'>进入用户登陆页面</router-link>          <router-link to='/user/register?name=jerry&id=321'>进入用户注册页面</router-link>          <div><router-view></router-view></div>      </div>  </template>  <script>  export default {      name: 'user'  }  </script> |

vue\_demo\src\components\routers\userLogin.vue

|  |
| --- |
| <template>      <div>这是用户登陆页面。。。。{{$route.params.name}}-{{$route.params.id}}登陆了</div>  </template>  <script>  export default {      name: 'userLogin'  }  </script> |

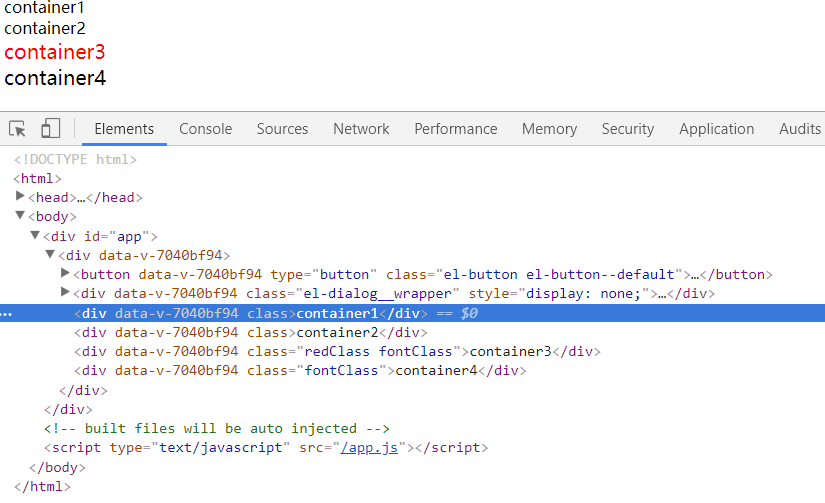
vue\_demo\src\components\routers\userRegister.vue

|  |
| --- |
| <template>      <div>这是用户注册页面。。。{{$route.query.name}}-{{$route.query.id}}想要注册</div>  </template>  <script>  export default {      name: 'userRegister'  }  </script> |

JS基础

1. Vue的Style写法
2. :class=”{‘class名’: data值}”
3. :class=”classObject”
4. :class=”[‘class1’, ‘class2’]”
5. :class=”[data? ‘class1’: ‘’, ‘class2’]”

|  |
| --- |
| <template>      <div>          <el-button @click="visible = true">Button</el-button>          <el-dialog :visible.sync="visible" title="Hello world">              <p>Try Element</p>          </el-dialog>          <div :class="{redClass: show}">container1</div>          <div :class="showClass">container2</div>          <div :class="['redClass', 'fontClass']">container3</div>          <div :class="[show? 'redClass' : '', 'fontClass']">container4</div>      </div>  </template>  <script>  export default {      name: 'home',      data: function() {          return {              visible: false,              show: true,              showClass: {                  redClass: true              }          }      },      mounted: function () {          this.show = false;          this.showClass = {               redClass: this.show          }      }  }  </script>  <style scoped>  .redClass {      color: red;  }  .fontClass {      font-size: 20px;  }  </style> |



## Vuex

1．安装

D:\skillmap\vue\vue\_demo>npm install vuex –save

2．使用

① 新建store.js文件

import Vue from 'vue'

import Vuex from 'vuex'

Vue.use(Vuex);

export default new Vuex.Store({

    state: {

        count: 0

    },

    mutations: {

        increment (state){

            state.count ++

        }

    }

})

➁ main.js文件引入

import Vuex from 'vuex'

Vue.use(Vuex);

import store from '@/store'

var vm = new Vue({

  el: '#app',

  router,

  store,

  components: { App },

  template: '<App/>'

})

➂　RightMain.vue

 <div>{{$store.state.count}}</div>

        <el-button @click="add">升高</el-button>

methods: {

        add() {

            this.$store.commit('increment');

        }

    }

1. 计算属性

export default new Vuex.Store({

    state: {

        count: 0

    },

    getters: {

        doubleCount(state){

            return state.count \*2

        }

    },

    mutations: {

        increment (state){

            state.count ++

        }

    }

})

 <div>{{$store.getters.doubleCount}}</div>

# Ui库使用

## 1．vue-cli引入bootstrap

npm install jquery –save

npm install bootstrap –save

npm install popper.js –save

webpack.base.config.js最后添加

plugins: [

new webpack.ProvidePlugin({

$: "jquery",

jQuery: "jquery"

})

]

主文件在文件头加入文件引入代码

import $ from 'jquery'

import 'bootstrap/dist/css/bootstrap.min.css'

import 'bootstrap/dist/js/bootstrap.min'

## 2．Element ui

**需求** 实现下面user追加和删除

### 4-2-1 安装

**Step1** >npm i element-ui –S

**Step2** vue\_demo\src\main.js中引入文件

**import** ElementUI **from 'element-ui'**;  
**import 'element-ui/lib/theme-chalk/index.css'**;  
Vue.use(ElementUI);

### 4-2-2 案例1



**Step3** vue\_demo\src\components\routers\userLogin.vue

|  |
| --- |
| <template>      <!-- <div>这是用户登陆页面。。。。{{$route.params.name}}-{{$route.params.id}}登陆了</div> -->      <div style="padding-top:20px">      <el-form ref="ruleForm" label-width="100px" class="demo-ruleForm">           <el-form-item label="名称" prop="name">              <el-input v-model="ruleForm.name"></el-input>          </el-form-item>          <el-form-item label="市内" prop="delivery">              <el-switch v-model="ruleForm.delivery"></el-switch>          </el-form-item>          <el-form-item>              <el-button type="primary" @click="submitForm()">创建</el-button>              <el-button @click="resetForm()">重置</el-button>          </el-form-item>      </el-form>      <el-table :data="userList" style="width: 100%; padding-left:50px">        <el-table-column prop="name" label="姓名" width="180"></el-table-column>        <el-table-column prop="delivery" label="市内人?" width="180"></el-table-column>        <el-table-column label="操作" width="100">          <template slot-scope="scope">              <el-button @click="deleteUser(scope.row)" type="text" size="small">删除</el-button>          </template>      </el-table-column>      </el-table>      </div>  </template>  <script>  export default {      name: 'userLogin',      data: function(){          return {              ruleForm: {                  name: '',                  delivery: false              },              userList: []          }      },      methods: {          submitForm: function (userList) {              var cgdRuleForm = {};              Object.keys(this.ruleForm).forEach((key)=>{                  cgdRuleForm[key] = typeof this.ruleForm[key] === 'boolean' ? this.ruleForm[key] === true ? '市内人': '不是市内人' : this.ruleForm[key]              })              this.userList.push(cgdRuleForm);              this.ruleForm = {                  name: '',                  delivery: false              }          },          resetForm: function () {              this.ruleForm = {                  name: '',                  delivery: false              }          },          deleteUser: function (dUser) {              var dIndex;              this.userList.forEach((user, index) => {                  if (dUser.name === user.name) {                      dIndex = index;                  }              });              (typeof dIndex === 'number') && this.userList.splice(dIndex,1);          }      }  }  </script> |

### 4-2-3 Layout布局

①<el-col :span="24"><div class="grid-content bg-purple-dark"></div></el-col>

传入的不是字符串就用:

➁　col之间存在间隔

<el-row :gutter="20">

<el-col :span="6"><div></div></el-col>

<el-col :span="6"><div></div></el-col>

<el-col :span="6"><div></div></el-col>

<el-col :span="6"><div></div></el-col>

</el-row>

### 4-2-4 对话框

* 1. 点击空白处无处理close-on-click-modal
  2. Visible.sync 父子组件同步，即点击X—》关闭dialog

也可以去除sync,将代码改为：

handleClose(done) {

            this.$confirm('确认关闭？')

            .then(\_ => {

                this.dialogVisible = false;

                done();

            })

            .catch(\_ => {});

      }

# 示例



* 1. Name、Ageを入力し、追加をクリックすることまたはEnterでUserをテーブルに追加する。
  2. SearchをFocus

注意点：

* 1. ElementUI的按键处理要加native

<el-col :span="5"><el-input placeholder="请输入姓名" v-model="name" @keyup.enter.native="addUser($event)"></el-input></el-col>

<el-col :span="6"><el-input placeholder="请输入过滤项" v-model="search" v-focus></el-input></el-col>

➁自定义指令，在inserted（被绑定元素插入父节点时调用）钩子中才有效

注意el是不是目标元素，防止绑定一直无效

<el-col :span="6"><el-input placeholder="请输入过滤项" v-model="search" v-focus></el-input></el-col>

directives: {

        focus: {

            inserted: function(el) {

                el.children[0].focus();

            }

        }

    }

完整代码：

<template>

    <div style="width:80%;margin: 0 auto">

        <div style="padding-top:20px">

            <el-divider></el-divider>

        </div>

        <!-- <div>这是用户注册页面。。。{{$route.query.name}}-{{$route.query.id}}想要注册</div> -->

        <el-row :gutter="20">

            <el-col :span="5"><el-input placeholder="请输入姓名" v-model="name" @keyup.enter.native="addUser($event)"></el-input></el-col>

            <el-col :span="4"><el-input-number v-model="age" :min="1" :max="100"></el-input-number></el-col>

            <el-col :span="6"><el-button type="primary" :disabled="name.length<=0" @click.stop="addUser">追加</el-button></el-col>

            <el-col :span="6"><el-input placeholder="请输入过滤项" v-model="search" v-focus></el-input></el-col>

        </el-row>

        <el-table :data="getUserList()" style="width: 100%" border highlight-current-row>

            <el-table-column prop="id" label="编号" width="100px"></el-table-column>

            <el-table-column prop="name" label="姓名" width="375px"></el-table-column>

            <el-table-column prop="age" label="年龄" width="475px"></el-table-column>

            <el-table-column label="操作" width="433px">

                <template slot-scope="scope">

                    <el-dialog title="确认删除" width="30%" :visible="dialogVisible">

                        <span>确认删除数据:{{dID}}</span>

                         <span slot="footer" class="dialog-footer">

                            <el-button @click="cancel">取 消</el-button>

                            <el-button type="primary" @click="confirm">确 定</el-button>

                        </span>

                    </el-dialog>

                    <el-button @click.prevent="deleteRow(scope.row.id)" type="text" size="small">移除</el-button>

                </template>

            </el-table-column>

        </el-table>

    </div>

</template>

<script>

export default {

    name: 'userRegister2',

    data() {

        return {

            name: '', //姓名框

            age: 1, //年龄框

            search: '', //检索框

            maxID: 0, //当前表格使用到的最大的ID

            userList: [], //user表格数据

            dID: 0,

            dialogVisible: false //删除数据弹出的确认框显示与否

        }

    },

    methods: {

        //user表格数据的获取方法

        getUserList() {

            if(this.search === '') {

                return this.userList;

            }

            return this.userList.filter(item => {

                return item.name.includes(this.search) || (new String(item.age)).includes(this.search);

            })

        },

        //追加用户

        addUser(e) {

            //enter键，name为空的话，不做任何处理

            if(e.key === "Enter" && this.name === ""){

                return;

            }

            this.getMaxID();

            this.userList.push({

                id: this.maxID + 1,

                name: this.name,

                age: this.age

            });

            this.name = '';

            this.age = 1;

            this.getUserList();

        },

        //“移除”的响应事件

        deleteRow(dID) {

            this.dialogVisible = true;

            this.dID = dID;

        },

        //删除数据

        delete() {

            return this.userList.some((item, index) => {

                if(this.dID === item.id){

                    this.userList.splice(index, 1);

                    return;

                }

            })

        },

        //获取表格最大的id

        getMaxID() {

            var ids = [];

            if(this.userList.length === 0) {

                this.maxID = 0;

                return;

            }

            this.userList.forEach(item => {

                ids.push(item.id);

            })

            this.maxID = ids.reverse()[0];

        },

        //弹框的cancel

        cancel() {

            this.dialogVisible = false;

        },

        //弹框的删除

        confirm() {

            this.dialogVisible = false;

            this.delete();

        }

    },

    directives: {

        focus: {

            inserted: function(el) {

                el.children[0].focus();

            }

        }

    }

}

</script>

<style scoped>

.el-table {

    margin-top: 10px;

}

</style>

# Others

## Li去除黑点

List-style-type: none

## DIV水平显示

自适应，满屏展示

1. <div class="box">
2. <div class="item"></div>
3. <div class="item"></div>
4. <div class="item"></div>
5. <div class="item"></div>
6. </div>
7. .box {
8. display: flex;
9. justify-content: center;
10. }
11. .item {
12. height: 30px;
13. flex: 1;
14. margin-right: 5px;
15. background: gray;
16. }

## 数组的基本操作

|  |
| --- |
| var arr = [1,3,4]  arr.pop() //尾部  arr.shift() //头部 |