Vue
Vue CLI
create vue
├ src/
│ ├ main.js
│ ├ App.vue
│ ├ components/
│ │ └ XXX.vue
├ public/
│ ├ index.html
│ └ style.css
Components 組件
<template></template>
<script>
export default { ... }
</script>
<style></style>
main.js
import App from "./App.vue";
import FriendConact from "./components/FriendConact.vue";
import 組件進來
const app = createApp(App);
app.component("friend-conact", FriendConact);
父組件(App)組上子組件(FriendConact)
app.mount("#app");
掛載到 html
父組件中使用子組件
<friend-conact></friend-conact>
在自定義組件上使用 v-for時，必須加上:key
在自定義組件上使用原生 JS事件，此事件將預設掛在子組件的根元素上
在 html中以標籤的屬性傳遞資料
<friend-conact phone-number="..."></friend-conact>
<friend-conact
:name="my.name"
:phone="my.phone"
></friend-conact>
若多項屬性名與標籤屬性名稱相同，可以 v-bind綁整個物件進去
<friend-conact
v-bind="my"
></friend-conact>
my = { name: "...", phone: "..." }
父組件
命名
kebab-case
以標籤的屬性方式命名
prop
prop: ["phoneNumber", "..."]
以 prop接收，接收後使用與 data的變數相同
子組件
命名
camelCase
JS變數不能使用"-"，Vue自動改變名稱為駝峰
驗證型態
props: { name: String, phone: String, email: String },
驗證
props: {
name: { type: String, required: false, ... },
email: { type: String, ...},
},
進階
type: String,
型態驗證
required: false,
必填驗證
default: "--",
預設值
validator: function (value) {
return value.indexOf("@") >= 0;
},
輸入驗證
驗證結果若有誤，會顯示在console
prop & emit
<friend-conact @toggle-favorite="toggleFavorite"></friend-conact>
function toggleFavorite() { ... }
父組件
命名
camelCase
this.$emit("toggle-favorite")
this.$emit
子組件
kebeb-case
事件名稱
命名
emit
父層用provide傳遞資料給予子孫組件使用
provide: { topic: ... }
使用如同 data，但若要傳送的資料依賴於 data，可寫成方法直接取用 data資料
provide() {
return { topic: this.topic }
}
provide
改變的必須是原值，即改變地址的方式將失去響應
響應性
message: computed(() => this.message)
可使用computed()函數提供一個計算屬性，保持響應性關係
子孫組件以 inject選擇接收哪些資料
inject: [...]
使用如同 prop
inject
功用如同 prop，但資料可以跨越多層組件向下傳，不限制於子組件才能使用，孫組件也能接收到
provide & inject
<Transition>
<TransitionGroup>
<keep-alive>
<component :is="goalsControl"></component>
</keep-alive>
使組件在切換時不完全刪除，保留組件狀態
<KeepAlive>
<teleport to="css選擇器">...</teleport>
將內容渲染到 DOM 中的另一個位置
傳送
<teleport>
<Suspense>
內置組件
類似組件的特性，但它們並非真正的組件，同時在模板編譯期間會被編譯掉。
取代動態的組件切換，:is=""動態帶入組件名稱
元組件
<component>
<component :is="goalsControl"></component>
<slot></slot>
預設/默認內容
出口
<slot>
<h2>Default</h2>
</slot>
<slot name="header"></slot>
對應插槽導入內容
具名插槽
<template v-slot:header>
簡寫:<template #header>
<header v-if="$slots.header">
<slot name="header"></slot>
</header>
插槽內容出口
<slot>
<slot goal-info="..."></slot>
訪問到子組件的狀態
<course-goal>
<template #default="courseGoal">
{{ courseGoal.goalInfo }}
</template>
</course-goal>
<course-goal #default="courseGoal">
{{ courseGoal.goalInfo }}
</course-goal>
特殊元素
父組件的樣式將不會滲透到子組件中
:deep()
在範圍樣式中，可影響到子組件
明確地將插槽內容作為選擇器的目標
:slotted
默認情況下，範圍樣式不會影響到 <slot/> 渲染出來的內容，因為它們被認為是父組件所持有並傳遞進來的
讓樣式規則應用到全局
:global
範圍樣式 scoped
樣式 style
app.components("friend-contact", {
template: `...`,
data() { ... },
...
});
Components 組件
kebeb-case
兩個單詞以上：html原有標籤皆為單一詞，可避免與 html原有標籤重複
命名
第一項參數 – 組件名稱
第二項參數即為一個Vue app
組件性質為隸屬於另一個 Vue app的 Vue app
組件是為其他Vue app所使用，所以不會掛載到 html上(沒有.mount("..."))，所以需要自帶模板：template
more in Vue CLI
Vue開發的
Router 路由
npm i vue-router@next
import { createRouter, createWebHistory } from "vue-router";
main.js
註冊
createWebHistory()
createWebHashHistory()
RouterHistory
history: RouterHistory
path: '/users'
動態路徑
path: '/users/:id'
url路徑
path
.* : 為一個匹配所有的正規表達式
AAA : 自訂名稱
path: '/:AAA(.*)*'
匹配所有路徑
作為輸入錯誤網址的顯示頁面，匹配所有未設定的網址，設置優先權應為最低，掛在routes的最後
用做辨識，防止在導航的 url中出現打字錯誤
name
對應的組件
component
default: TeamsList,
footer: TeamsFooter
多個不同的 router-view以屬性 name區分，配置對應組件
components
設為true，將動態路徑的參數在組件中以 props取用，取代以$route查找 url路徑上的參數
props
改變 url
重新定向
redirect
表示不同路徑會載入相同的組件
別名
alias
參數與routes一樣為：RouteRecordRaw[]
children
使用守衛
beforeEnter(to, from, next) {...}
RouteRecordRaw[]
路由配置
routes: RouteRecordRaw[]
const router = createRouter({RouterOptions});
RouterOptions
替換路徑中<router-link>上的class名稱，預設為："router-link-active"
linkActiveClass: "String"
替換精準路徑<router-link>上的class名稱，預設為："router-link-exact-active"
linkExactActiveClass: "String"
to: 到哪
from: 從哪
參數
{left: 0, top: 100}
savedPosition: 只有按上下頁時有值，紀錄頁面滾動到的位置
滾動行為
JS物件
ScrollToOptions
return ScrollToOptions
scrollBehavior(to, from, savedPosition) {...}
https://router.vuejs.org/zh/guide/advanced/scroll-behavior.html
app.use(router);
模仿 window.history的API操作
留有歷史紀錄的導航
this.$router.push(...)
router.push(...)
編程式
同router.push，但為覆蓋紀錄，未保留前個歷史紀錄
<router-link :to="..." replace>
聲明式
替換現在位置
router.replace(...)
編程式
router.forward()
往前
router.back()
返回
router.go(n)
橫跨
跳轉歷史
n：Number 移動的頁數，若數字大於可移動紀錄，為失敗
正數：往前，負數：返回
<router-link :to="/team/11">
帶入url
導航參數
name: 'TEAMS'
name
參考 > 路由配置 > name
鏈接到一個命名的路由
params: { teamId: '11' }
params
參考 > 路由配置 > path、props
物件可帶上動態路徑的參數
帶入物件
query: { sort: "asc" }
加入查詢參數
query
在導航進子組件頁面後，可以看到在網址後多了"?sort=asc"
傳遞方式為存在url上
不是用 props，子組件使用時不能直接當屬性用(用 this取用)
this.$route.query
子組件取用
與 params比較
不受限於路由配置，使用較自由
優點
為可選的參數，使用時不用另外在路由配置設定
子組件就要好缺此參數的準備
缺點
訪問路由 $router
路由導航
路由參數
params
全路徑
path
路徑 $route
$
參考 > 路由配置 > path、name、component、components、children
顯示與 url對應的組件
<router-view></router-view>
多個可渲染位置做命名區分
屬性name
參考 > 路由配置 > name
https://router.vuejs.org/zh/guide/essentials/nested-routes.html
嵌套路由
守衛是異步執行的，導航將等待守衛 resolve完
router.beforeEach((to, form [, next]) => {...})
to
form
next
接收參數
可選參數，為一個函數
直接 return
調用第三個參數 next()
注：return控制為新版的 vue router文件所建議使用
寫法
導航控制
註冊一個全局導航守衛
前往目標頁面
true
不傳參數(不回傳值)
前往目標頁面
導航有效
參考 > 路由導航 > 導航參數
前往指定頁面，參數同使用router.push()時的參數一樣
路由地址
{ name: "TEAM_MEMBERS", params: { teamId: "t1" } }
導航控制成功/失效傳的參數
false
取消導航
因導航成功後會再次調用守衛，須注意避免無限重新定向
參考 > 路由配置 > beforeEnter
在路由配置中指定路由使用守衛
https://router.vuejs.org/zh/guide/advanced/navigation-guards.html
導航守衛
運作原理
proxy
設置 Handler，使資料建成動態更新的效果
ref
在 html中註冊：ref
<input type="text" ref="userName">
在 vue.js中調用使用：$refs
this.message = this.$refs.userName.value
用法和v-model很像，v-model是監測每一次的資料變動，ref是在要使用時才去抓資料
ref與原生JS存取的內容都是 String
生命週期
beforeCreate()
created()
beforeMount()
mounted()
beforeUpdate()
update()
beforeUnmount()
unmounted()
https://cn.vuejs.org/guide/essentials/lifecycle.html
https://cn.vuejs.org/api/options-lifecycle.html
v-text v-html v-show v-slot v-pre v-memo v-cloak
v-if v-else v-else-if
v-model
雙向綁定
修飾符
.number
將輸入的合法符串轉為數字
.lazy
監聽 change 事件而不是input
.trim
移除輸入內容兩端空格
若是從 type為 number的 input抓回資料
v-model無需修飾符，值也將自動轉為 Number
v-once
僅渲染元素和組件一次，並跳過之後的更新
v-on 事件處理
@click="add()"
簡參數時，有無括號是相同的
$event
事件參數保留字
修飾符
事件修飾符
.stop
event.stopPropagation()
.prevent
event.preventDefault()
.self
當 event.target是元素本身時才觸發，ex:click點在子元素上無效
.capture
指向內部元素的事件，在被內部元素處理前，先被外部處理
.once
最多觸發一次
.passive
將立即發生而非等待
不可與.prevent一起使用
按鍵修飾符
常用按鍵
.enter .tab .esc .space .up .down .left .right
.delete
Delete & Backspace
系統按鍵
.ctrl .alt .shift
.meta
Mac
Command 鍵(⌘)
Windows
Windows 鍵(⊞)
奇怪系統奇怪鍵盤
Sun 微機系統？這啥？
鑽石鍵(◆)
MIT 和Lisp 機器
Knight 鍵盤
space-cadet 鍵盤
Symbolics 鍵盤
與 keyup 事件一起使用時，系統按鍵必須在事件發出時處於按下狀態
@keyup.ctrl
在鬆開了另一個鍵時被觸發，單獨鬆開 ctrl鍵將不會觸發
.exact
觸發確定組合的系統按鍵修飾符
@click.ctrl
按下 Ctrl 時，即使同時按下 Alt 或 Shift也會觸發
@click.ctrl.exact
僅按下 Ctrl且未按任何其他鍵
@click.exact
沒有按下任何系統按鍵
鼠標按鍵
.left .right .middle
指令 Directives
v- 前綴內置指令
v-for
可綁定目標
物件
v-for="(value, name, index) in object"
陣列
v-for="(value, index) in array"
數字
v-for="count in 10"
字串
v-for="letter in 'Hello world!' "
iterable？？？
"綁定目標"有資料變動時，vue默認做的是"更新"而非"移動"Dom，所以若元素中有未綁定在"綁定目標"上的資料就不會有更動，使用 key可以強制作元素排序，將元素中所有資料一同移動

{{ item.text }}
<input type="text">

input的資料未與 items有綁定，當 items有更動時此資料將被留在原地
可以直接將 input資料新增存進 items做綁定，但此資料可能並不是 items需要的，或其他原因不能對目標做控制，就需要用到 key值

{{ item.text }}
<input type="text" :value="item.text">

:key="
<div v-for="item in items" :key="item.id">
{{ item.text }}
<input type="text">

v-bind Attribute綁定
如果綁定的值是 null 或者 undefined，那麼該 attribute 將會從渲染的元素上移除。
布林值
決定attribute 是否應該存在於該元素上
:disabled="true"
動態綁定多個值
多個 attribute的 Object，不帶參數使用 v-bind="..."綁定
objectOfAttrs: { id: 'container', class: 'wrapper' }
v-bind="objectOfAttrs"
參數 Arguments
某些指令會需要一個"參數"
v-bind
:href="..." ，href是參數
v-on
@click="..." ，click是參數
動態參數
{attributeName}="..."
@[eventName]="..."
動態參數的限制
表達式的值只能為 string，或者是 null
null 將移除該綁定，其他非 string的值都會 error
表達式不得有空白或引號
若表達式複雜改傳入"計算屬性(data or computed)"
傳入的"計算屬性(data or computed)"名稱不該有大寫
因為瀏覽器會強制將其轉換為小寫，與組件寫的名稱不同會失效
使用DOM 內嵌模板(直接寫在HTML文件裡的模板)會有此限制，單文件組件內的模板不受此限制，但還是都用小寫好...
修飾符 Modifiers
可以使用鏈式書寫
@click.stop.prevent="..."
需要注意調用順序
@click.prevent.self
會阻止"元素及其子元素"的所有點擊事件的默認行為
@click.self.prevent
只會阻止對"元素本身"的點擊事件的默認行為
可以只有修飾符
@click.prevent
狀態選項
data
watch
也許用在設置計時器、發送Http請求、儲存storage，當有數據更動時在背景做處理
不在template中使用
computed
沒有 input；有 return；使用時實際為呼叫一個函數，但是使用方法如同"data"一樣，不要括號
Vue document：
若我們將同樣的函數定義為一個方法而不是計算屬性，兩種方式在結果上確實是完全相同的，
然而，不同之處在於計算屬性值會基於其響應式依賴被緩存。
一個計算屬性僅會在其響應式依賴更新時才重新計算。
當重渲染時：
用 method中的函式 => 執行
用 computed => 若變動的非依賴項，即抓緩存中的計算結果，資料不更新
減少不必要的執行工作對效能較好
methods
props
emits
expose
不得用於事件綁定
文本插值
可以單一表達式或指定函數
單一表達式
可以...
data
{{ item }}
算式
{{ 1 + 2 }}
Math.random()
{{ Math.random() }}
三元表達式
{{ true ? "X" : "O" }}
不可以...
var a = 3
三元表達式以外的維作控制
if { ... } ... else ...
可用是否能放在 return後當判斷
指定函數
methods的 function
{{ add(2, 3) }}
純文本
v-text="..."
{{ ... }}
innerHTML
v-html="..."
動態渲染任意HTML 是非常危險的，容易造成 XSS 漏洞，僅在內容安全可信時再使用v-html
Http
FireBase
Google產品
gmail 登入
Realtime Database 即時資料庫
建立後最上方生成api網址
瀏覽器內建方法
發送 Http請求方法
fetch
Request
fetch('url', {info})
url
帶入 FireBase生成的網址
https://learn-vue-http-285cc-default-rtdb.firebaseio.com/
網址後接上一個 json檔名作為儲存位置辨識
https://learn-vue-http-285cc-default-rtdb.firebaseio.com/survey.json
Info
method 請求方法
"POST"
"GET"
"DELETE"
headers
"Content-Type" 資料格式說明
"application/json"
body 傳送資料主體
需符合 headers描述的資料
JSON格式資料轉換
轉成JSON格式：JSON.stringify
JSON格式轉回：JSON.parse
Response
返回 Promise
可使用 then()、catch()和 async/await
Axios
第三方：額外安裝
https://github.com/axios/axios
import axios from 'axios';
Request
axios.post('url', {info});
url
info
傳送資料主體
自動將正文數據轉換為 JSON
Response
返回 Promise
可使用 then()、catch()和 async/await
error Catch
.catch((error)=>{...})