Requisitos

Node 16.17

Pasos:

TIPS:

* Propiedades y métodos propios del sistema se identifican con $ para diferenciarlos de nuestro métodos y propiedades
  + $root , nos da acceso a todo el objeto
  + $data -> la carga de datos en si
  + $el => elemento html
  + // $watch es un método de instancia  
    vm.$watch('a', function (newValue, oldValue) {  
     // Esta función será ejecutada cuando `vm.a` cambie  
    })
* Interpolacion ( Modo Mustache ) {{}}
* Para ver html puro en un template ( directiva v-html )
  + <p>Using mustaches: {{ rawHtml }}</p>  
    <p>Using v-html directive: <span v-html="rawHtml"></span></p>
* v-bind enlaza un atributo en el template ( método abreviado : )
* v-on enlaza un evento en el template ( método abreviado @ )
* La propiedad data de la instancia Vue implica la reactividad , todo cambio en data produce un cambio en objeto (solo las inicializadas a origen )
  + Ojo, si se incorporan con posterioridad a data no implica cambio ( por eso hay que inicializarlas en algo)
  + Ojo: en las mutaciones de array ( tipo push, clear, etc, ) , Vue no se “entera” de una asignación directa tipo aDatos[3] = “valor”
    - Si fuera necesario , lo ideal seria usar el método
      * Vue.set(vm.items, indexOfItem, newValue)
* v-slot es similar al children en react , reemplaza el contenido en la marca slot
* Componentes dinámicos con la propiedad v-bind:is
  + <!-- El componente cambia cuando currentTabComponent cambia -->  
    <component v-bind:is="currentTabComponent"></component>

SESION 1

TEMA 1 ( INSTALACION LOCAL VUE )

TEMA 2,3,4 MiniDemos en local

( INSTALACION NPM ), explicación proyecto

TEMA 5 Componentes

TEMA 1. Instalación Local Vue

* mkdir c:\Vue2\01\_local
* Creamos el index.html

<html>

<head>

<script src="https://cdn.jsdelivr.net/npm/vue@2.7.16/dist/vue.js"></script>

</head>

<body>

<div id="app">

<p>{{ message }}</p>

</div>

<script>

console.log("hola vue" , Vue)

new Vue({

el: '#app',

data: {

message: 'Hola Mundo!'

}

})

</script>

</body>

</html>

* Lo vemos desde el navegador
  + Explicamos el CDN y los problemas de versión
  + Tips
    - Descarga del cdn en local
    - Ver index2 para usar templates modo local

TEMA 0. Instalación del entorno

* mkdir c:\Vue2\
* npm install -D vue@^2 ( 2.6.14 )
* npm i -D @vue/cli
* Revisamos la instalación
  + Npx vue –version ( 5.0.8 )

TEMA 1. Instalación Mediante VUE CLI

* npx vue create 01\_vuecli ( elegimos Vue2/Eslint)
* cd 01\_vuecli
* npm run serve
* Analizamos contenido
  + Package.json , jsconfig.json, babel.config.json
  + Public/index.html ( Cargador de nuestra app )
    - Las macrosustituciones del html ( algo complejo , las variables del sistema en trasnpolacion <%%>
    - https://stackoverflow.com/questions/62023604/where-to-find-or-how-to-set-htmlwebpackplugin-options-title-in-project-created-w
    - <div id="app"></div>
    - Como siempre , si necesitamos algo mas lo ponemos aqui
  + Src/assets ( Ponemos nuestros ficheros fijos , gráficos , etc )
  + Main.js ( cargador principal de la app )
    - Si necesitamos ampliar las instancias irían aquí
  + App.vue ( Primer componente ) con su estructura
    - Template
      * Parte visual y DOM
    - Script
      * Programacion
    - Style
      * estilo
  + Para añadir componentes , lo necesito hacer en imports del script )
* PRIMERA PRUEBA ( hola mundo )
  + Copiamos el componente helloworkd.vue a holamundo.vue
  + Cambiamos el html
  + En la clase ( name = “selector” , props:”parámetros” )
  + Atributo “data” , la reactividad de los datos

Explicando componentes

* package.json: Analizamos el proyecto

TEMA 3. Renderizado

Directriz condicional

Podemos usar template para condicionar todo un bloque

<h1 v-if="ok">Sí</h1>  
<h1 v-else>No</h1>

<template v-if="ok">  
 <h1>Título</h1>  
 <p>Párrafo 1</p>  
 <p>Párrafo 2</p>  
</template>

Directriz v-show

Visualizar / ocultar , difiere de v-if porque si se genera , solo lo oculta

Directriz v-for , renderizado de lista

Podemos acceder al objeto de la colección y al indice

<li v-for="(item, index) in items">  
 {{ MensajePadre }} - {{ index }} - {{ item.mensaje }}  
 </li>

In , recorremos propiedades Object.keys(obj)

Of: recorremos elementos array

Uso del key para evitar duplicidades ( el DOM lo usa para optimizar )

TEMA 5. Creación componente

Vamos a hacer un grid con un array de contadores , cada contador tiene su independencia , start , stop , tick , etc

* Cd c:\Vue2
* npx vue create 05\_componentes ( Vue2 , Eslint )
* Creamos dos componentes ( simplemente copiando el helloworld.vue )
  + Header.vue ( Cabecera genérica )
    - <template>
    - <h1>Cabecera</h1>
    - </template>
    - <script>
    - export default {
    - name: 'MiCabecera',
    - }
    - </script>
    - <style scoped>
    - h1 {background-color: aquamarine;}
    - </style>
  + Contador.vue ( Componente unitario )
* <template>
* <div>
* <h4>Titulo {{ titulo }}</h4>
* <h5>Valor {{ valor }}</h5>
* <button v-on:click="toggle()">{{lRun?'Parar':'Arrancar'}} </button>
* <button @click="resetear()">Resetea</button>
* </div>
* </template>
* <script>
* export default {
* name: 'MiContador',
* props:['titulo','inicio'],
* data(){
* return{
* valor: 0,
* lRun:true
* }
* },
* mounted(){
* this.timer = setInterval(this.tick.bind(this), 1000)
* this.valor =  this.inicio ?? 0;
* },
* beforeDestroy() {
* this.destroy()
* },
* methods:{
* destroy () {
* clearInterval(this.timer)
* },
* tick(){
* if(this.lRun){
* console.log("this.valor", this.valor)
* this.valor++;
* }
* },
* toggle(){
* this.lRun = !this.lRun;
* },
* resetear(){
* this.valor =  this.inicio;
* this.$emit('escucho\_msg', this.valor);
* console.log("emito", this.$emit)
* }
* }
* }
* </script>
* <style scoped>
* h4 {color: burlywood;}
* h5 {color:darkblue}
* </style>
  + La pagina principal ( App.vue )

TEMA 5. Creación componente

* Cd c:\Vue2
* Npx vue créate 05\_componentes ( Vue2 , Eslint )
* Npm run serve
* En componentes creamos Header ( cojo helloworld.vue )
  + <template>
  + <h1>Cabecera</h1>
  + </template>
  + <script>
  + export default {
  + name: 'MiCabecera',
  + }
  + </script>
  + <style scoped>
  + h1 {background-color: aquamarine;}
  + </style>
* En componentes creamos Contador.vue vacio ( cojo hello)
  + <template>
  + <h1>Contador</h1>
  + </template>
  + <script>
  + export default {
  + name: 'ContadorComponent',
  + props: {}
  + }
  + </script>
  + <!-- Add "scoped" attribute to limit CSS to this component only -->
  + <style scoped>
  + </style>
* Adapto App.vue
  + Template
    - <template>
    - <div id="app">
    - <Cabecera></Cabecera>
    - <Contador></Contador>
    - </div>
    - </template>
  + Script
    - <script>
    - import Cabecera from './components/Header.vue'
    - import Contador from './components/Contador.vue'
    - export default {
    - name: 'App',
    - components: {
    - Cabecera,Contador
    - }
    - }
    - </script>
* Resultado dos h1
* Ponemos varios contadores
* Incorporamos un array de contadores
* Vemos la variable aContadores con {{ aContadores }}
* Creamos un table para ajustar los componentes y vemos estructura for
  + <table width="100%" border="1">
  + <thead>
  + <tr>
  + <th v-for="contador in aContadores" :key="contador.id">{{ contador.titulo }}</th>
  + </tr>
  + </thead>
  + <tbody>
  + <tr>
  + <td v-for="contador in aContadores" :key="contador.id"><Contador></Contador></td>
  + </tr>
  + </tbody>
  + <tfoot>
  + <tr>
  + <td v-for="contador in aContadores" :key="contador.id"><button>Algo</button></td>
  + </tr>
  + </tfoot>
  + </table>
* Ahora me interesa pasarle las propiedades ( en Contador con props )
* Le paso el valor inicial (inicio) del contador via atributo ( props )
* Incorporo el ciclo de vida del componente
  + <https://elabismodenull.wordpress.com/2017/05/05/vuejs-el-ciclo-de-vida-de-un-componente/>
  + beforeCreated, created, beforeMount , mounted, beforeupdate , update
  + activated , deactivated
* En mounted , inicializamos la propiedad inicio y comprobamos que actualiza valor con inicio
* En contador incorporo un botón + para incrementar a mano el valor

<button v-on:click="incrementaValor()">+</button>

* Y la creación del método en la clase
  + methods:{
  + incrementaValor(){
  + this.valor++;
  + }
  + }
* Compruebo que se mueve
* Incorporo un timer que cada segundo , ejecute el método tick
  + mounted(){
  + this.timer = setInterval(this.tick.bind(this), 1000)
  + this.valor =  this.inicio ?? 0;
  + },

TEMA 7. Uso avanzado de Componentes

* <https://medium.com/@dgongoragamboa/introducci%C3%B3n-a-layouts-slots-y-composables-en-vue-17f080e9e67d>

TEMA 8. Formularios

* v-model para binding (bidireccional)
  + <input v-model="message" placeholder="edíteme">  
    <p>El mensaje es: {{ message }}</p>
* Elementos , input , textarea , checkbox , radio, select
* Se pueden usar conversión del tipo con attributes tipo value

<input  
 type="checkbox"  
 v-model="toggle"  
 true-value="yes"  
 false-value="no"  
>

* Modificadores de filtro ( solo números , o sin espacios)
  + - v-model.number
    - <input v-model.trim="msg">

TEMA 9. Directivas

* Sistema de marcado de componentes “v-“ , el renderizador busca las directivas y evalua su contenido
* v-text Equivalente a {{ texto }}. Usa .textContent internamente.
* v-html Inserta HTML en un elemento sin procesarlo. Usa .innerHTML internamente.
* v-pre Mantiene las {{ templates }} del elemento intactas, sin renderizar.
* v-once Renderiza las {{ templates }} solo la primera vez, y no lo hace más.
* v-cloak Directiva que permanece hasta que la {{ template }} se renderiza con contenido.
* v-model Sí Enlaza el valor de una variable con un <input>, <select>, <textarea> o un componente.
* v-show Muestra/oculta el elemento, alternando con un display: none.
* v-if Equivalente a un if de Javascript. Acepta una expresión Javascript por parámetro.
* v-else-if Equivalente a un else if de Javascript. Acepta una expresión Javascript por parámetro.
* v-else Equivalente a un else de Javascript. No tiene parámetros.
* v-for Estructura for
* v-bind Enlace con data
* v-on Eventos
* Directivas globales

// Registra una directiva personalizada global llamada `v-focus`  
Vue.directive('focus', {  
 // Cuando el elemento enlazado se inserta en el DOM...  
 inserted: function (el) {  
 // Enfoca el elemento  
 el.focus()  
 }  
})

* Directivas locales en cfg

directives: {  
 focus: {  
 // Definición de directiva  
 inserted: function (el) {  
 el.focus()  
 }  
 }  
}

* Los Hooks en las directivas es similar al ciclo de vida de un componente
  + Bind, inserted, update, componentUpdated,unbind)

TEMA 10. Filtros

* Pipes en Angular
* Se define en la config de vue
* Pueden ser encadenados
* Y paso de argumentos

filters: {  
 capitalize: function (value) {  
 if (!value) return ''  
 value = value.toString()  
 return value.charAt(0).toUpperCase() + value.slice(1)  
 }  
}

* Los filtros pueden ser globales

Vue.filter('capitalize', function (value) {  
 if (!value) return ''  
 value = value.toString()  
 return value.charAt(0).toUpperCase() + value.slice(1)  
})

* Mixins ( La propiedad de “mezclar” un componente con otro)
  + Se fusionan las propiedades y métodos
    - Útil para componentes diversos con mismos métodos p.e.

TEMA 11. Transiciones

* <https://es.vuejs.org/v2/guide/transitions>

TEMA 17. Capacitor

<https://capacitorjs.com/docs/getting-started>

<https://capacitorjs.com/solution/vue>

Generamos un proyecto limpio ( vue créate … )

npm install @capacitor/core @capacitor/cli

npx cap init

npm run build

npm i @capacitor/ios @capacitor/Android

npx cap add Android

* Npx cap copy
* Npx cap sync
* Npx cap open Android

TEMA 18. Nuxt

// <https://github.com/nuxt/nuxt/issues/10844>

* Recomiendo yarn por los problemas
* Manual
  + {
  + "name": "my-app",
  + "scripts": {
  + "dev": "nuxt",
  + "build": "nuxt build",
  + "generate": "nuxt generate",
  + "start": "nuxt start"
  + }
* }
* yarn add nuxt
* creaamos c:\vue2\18\_nuxt\pages\index.vue
* npx nuxi dev
  + ¿Qué es NuxtJS?
    1. Nuxt es una plataforma web para servir paginas tanto dinámicas como estáticas
  + ¿Qué es el Renderizado desde Servidor?
    1. La posibilidad de que sea una petición remota quien genere la pagina y no el cliente
  + NuxtJS vs otros Renderizados desde Servidor
    1. Next , SEO , etc
  + Creando una app con NuxtJS
    1. yarn add nuxt
  + Análisis de la estructura de carpetas y ficheros
  + ¿Qué se puede crear con NuxtJS?
  + Las rutas en NuxtJS
  + Creando rutas con direcciones dinámicas
  + Añadiendo links y analizando la navegación
  + Validación de parámetros
  + Creación de rutas enlazadas
  + Creando un layout
  + Usando componentes con NuxtJS
  + El fichero de configuración de NuxtJS
  + Trabajando con variables de entorno
  + Configuración del enrutado de NuxtJS
  + Animando transiciones de página
  + Añadiendo plugins
  + Registrando un filtro de datos
  + Comprendiendo los módulos
  + Añadiendo un Middleware
  + Testando un Middleware
  + Comenzando un proyecto con una plantilla
  + Aplicaciones Universales vs SPA vs Estáticas
  + Construyendo una aplicación Universal
  + Construyendo una aplicación SPA
  + Construyendo una aplicación Estática
  + Mejorando el proceso de generación
  + Limitando la cantidad de requests
  + Ajustando el Store de Vuex
* npx nuxt@2.15.8 init 18\_nuxt ( Me falla )
* npm init nuxt-app@latest 18\_nuxt

TEMA 21. Proyecto

* En nuxt (manual )montamos el proyecto
* Mkdir c:\vue2\21\_proyecto
* Cd c:\vue2\21\_proyecto
* Npm add nuxt

C:\GUION\_Vue>npm init nuxt-app@2 21\_proyecto\_nuxt2

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Propiedades computadas:

Operaciones complejas ,repetitivas pueden incorporarse en una propiedad computada para usarse múltiples veces

Las propiedades computadas están “cacheadas” ,es decir no se reejecutan si data no cambia.

<div id="example">  
 <p>Mensaje original: "{{ message }}"</p>  
 <p>Mensaje invertido computado: "{{ reversedMessage }}"</p>  
</div>

var vm = new Vue({  
 el: '#example',  
 data: {  
 message: 'Hola'  
 },  
 computed: {  
 // un getter computado  
 reversedMessage: function () {  
 // `this` apunta a la instancia vm  
 return this.message.split('').reverse().join('')  
 }  
 }  
})

Bind clases CSS v-bind

Podemos pasar un objeto a v-bind:class para la asignación dinámica de clases:

<div class="static"  
 v-bind:class="{ active: isActive, 'text-danger': hasError }">  
</div>

data: {  
 isActive: true,  
 hasError: false  
}