Escuela Politécnica Superior de Alicante

Diseño de aplicaciones web

Intro a Vue.js

Contenidos

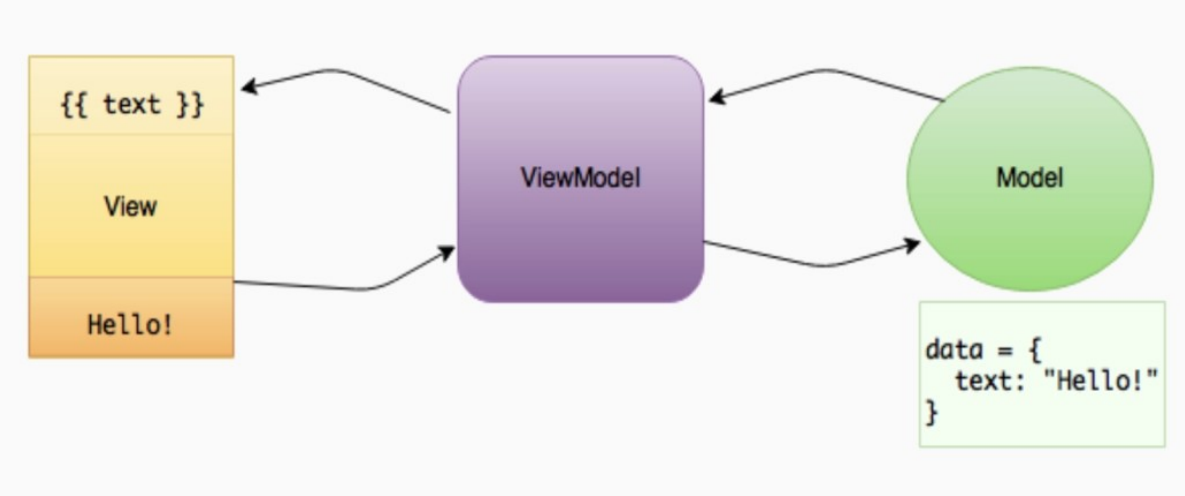
- ¿Que es Vue.js?
- ¿Cómo funciona?
- Características:
  - Directivas
  - Computed, watchers
  - Métodos
  - Filtros
- Formularios
- Componentes
- Otros módulos: axios, fetch, Vuex, routing, i18n

¿Que es Vue.js?

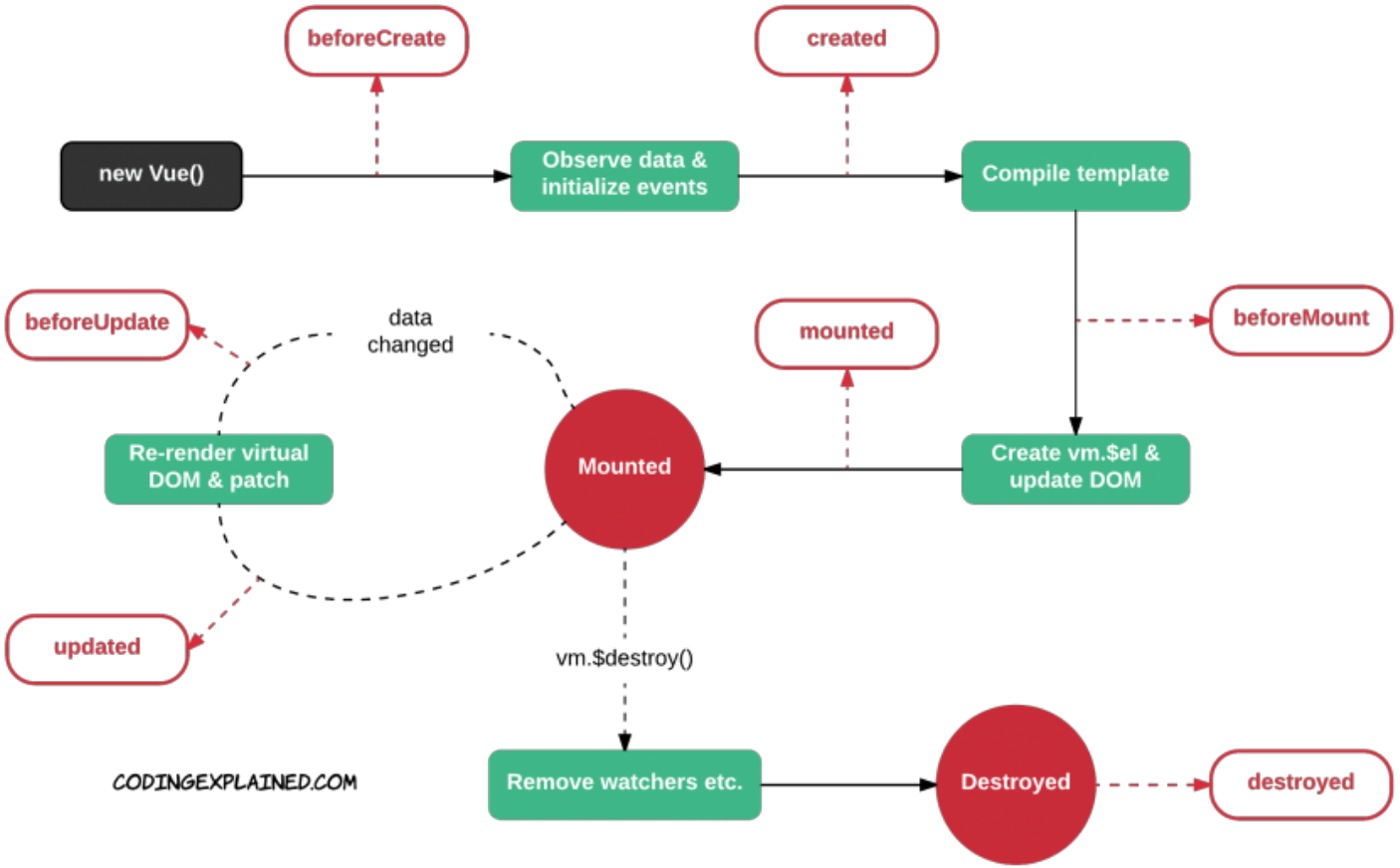
- Sitio web oficial: https://vuejs.org/
- Framework de cliente para crear Front-ends (Interfaces de usuario)
- Es un framework incremental/modular: carga solo los módulos que se necesitan.
- Intercepta el DOM de la página web para tomar el control del mismo (mantiene una copia del DOM en caché).
- Es compatible con otros frameworks, librerias y proyectos ya existentes.

¿Que es Vue.js?

- Usa el patrón MV-VM> Model-View-ViewModel.
- Se encarga de la capa intermedia entre los modelos y el UI.
- Supone un cambio de paradigma: se manipulan los modelos y el ViewModel ajustará automáticamente el DOM → **Programación reactiva**.



Ciclo de vida



¿Cómo funciona?

```
Var app = new Vue({
   el: ’#app’,
   data: {mensaje: ‘Hola mundo’}
})
```

- el: id de la capa donde se vincula la instancia de Vue.

```
<div id=“app”>
   {{mensaje}}
</div>
```

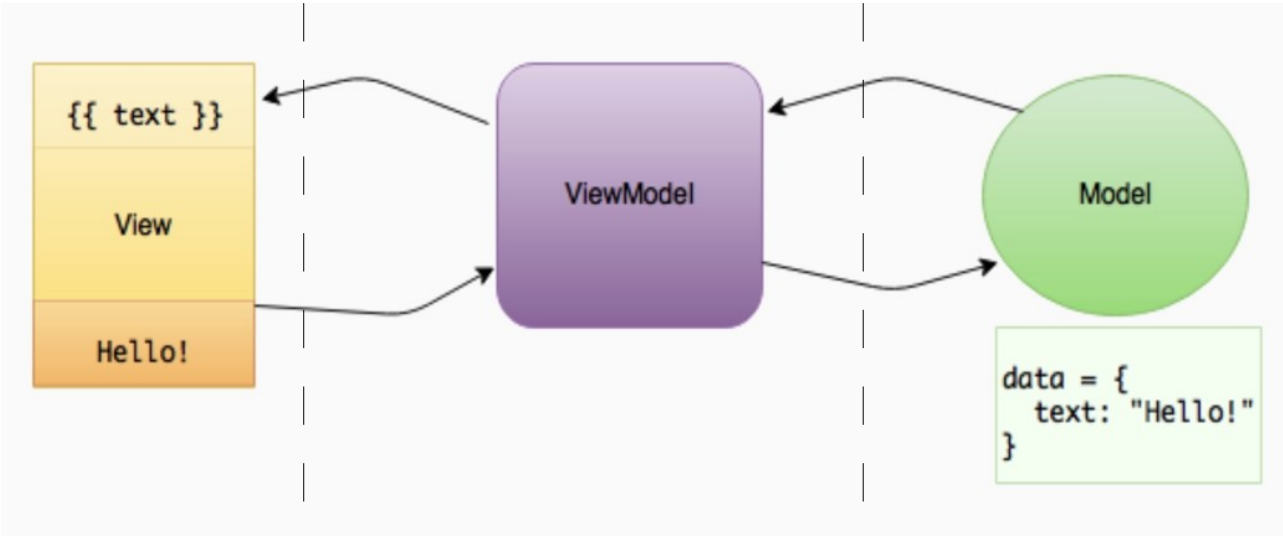
- Template: expresiones con formato 'moustache', que vuelcan contenido HTML desde la instancia Vue.

¿Cómo funciona?

```
<div id="app">
    {{mensaje}}
</div>
```

```
Var app = new Vue({
    el: '#app',
    data: modelo
})
```

```
var modelo = {
    mensaje: 'Hola'
}
```



Directivas

- Atributos especiales que interpreta Vue.
- Son reactivos, enlazados en dos direcciones.
- Prefijos: 'v-'
- v-bind:atributo=”expresión”
  - abreviatura: ':atributo'
  - Vincula un atributo a una expersión del modelo.
- V-on:evento=”manejador(...)”
  - atajo: @evento
  - Modificadores: .stop, .prevent, .once, ...
  - Ejemplo: v-on:click.prevent=”funcion()”

Directivas

- Directivas para sentencias de control:
- v-for=”item in array”, v-for=”(item, index)...”
  - Para iterar sobre un array o colección.
  - Requiere una clave: v-bind:key=”item.id”
- v-if=”expresion”
  - muestra o no un elemento en función de si la expresión se evalua a true.
- v-show: igual que v-if pero siempre renderiza el contenido.

Directivas

- Otras directivas:
  - v-else, v-else-if
  - v-html: para inyectar código HTML en un elemento.
  - v-model: para vincular controles de formulario con elementos del modelo.

Computed properties

- Permite encapsular el cálculo de expresiones complejas de forma óptima para luego usarlas en templates.
  - Son cacheadas y solo se recalculan si sus dependencias cambian.

```
computed:{
  datoCalculado: function(){
    return expresion_compleja;
  }
}
```

Watchers

- Establecen un triger que se activa cuando una variable se modifica.
  - Usados para cargar contenido dependiente de una variable o para realizar operaciones costosas en respuesta a un cambio.

```
watch:{
  pais: function(paisSeleccionado){
    this.cargarProvincias(paisSeleccionado);
  }
}
```

Métodos

- Funciones o métodos de la instancia o vinculadas a un objeto o un componente.
  - Pueden recibir argumentos en su invocación y devolver datos.
  - Se ejecutan siempre que se invoquen, al contrario que las 'computed'.

```
methods:{
  miFuncion: function(parametros){
    ...funcionalidad...
    return ...;
  }
}
```

Filtros

- Funciones que permiten aplicar formato a propiedades del modelo.
- Se aplican a:
  - Templates: `{{ dato | nombreFiltro }}`
  - Expresiones con v-bind.

```
filters:{
    nombreFiltro: function(){
        return ...;
        }
    }
```

Formularios

- Para enlazar campos de formularios con datos del modelo se usa: v-model.
- Se aplica a: inputs de texto, textareas, selects, y botones de radio y de check.
- v-model realiza un enlace en dos direcciones, por lo que equivale a usar:
  - v-bind:value → del modelo a la vista
  - v-on:input → de la vista al modelo
- En los botones de radio y de check se puede usar los modificadores true-value y false-value para indicar valores cuando esta activado o no.

Formularios

- Ejemplos:
  - Caja de texto:

```
<input type=”text” v-model=”cliente.nombre” />
```

  - Botones de radio

```
<input type=“radio” id=“one” value=“One”
v-model=“picked”>
<input type=“radio” id=“two” value=“Two”
v-model=“picked”>
```

  - Checkbutton:

```
<input type=“checkbox” v-model=“aprobado” true-
value=“Sí” false-value=“No”>
```

Componentes

- Permiten encapsular y reutilizar código de lógica de modelo y de representación → ViewModels.

Se deben declarar antes que la instancia Vue:

```
Vue.component('mi_componente',{
    template:`<div>Algo:{{dato}}</div>`,
    data: function(){ return{dato:10} },
    })
```

- Se referencian como un elemento HTML con su propia etiqueta:

```
<mi_componente></mi_componente>
```

Componentes

- Los componentes se pueden anidar jerárquicamente.
- Para comunicar un componente con su padre se usan propiedades que se declaran en el componente hijo:

```
Vue.component('hijo',{
    props: ['dato'],
    template: `<span>{{dato}}</span>`
})
```

- y se dan valor cuando el padre lo instancia:

```
<hijo dato="¡Hola!"></hijo>
```

Componentes

- Para que un componente hijo pueda comunicarse con su padre, se usan eventos que el hijo emite hacia su padre:

```
this.$emit('evento_hacia_el_padre')
```

- y el padre debe capturarlos con un manejador propio:

```
<comp_hijo v-on:evento="manejador"></comp_hijo>
```

Axios, fetch

- Si deseamos interactuar con un back-end (una API de servicios Rest) necesitaremos realizar peticiones http asíncronas.
- Con Vue.js podemos usar varias librerias de este tipo, las mas utilizadas: **Http, Axios y Fetch**
- Permiten realizar solicitudes http de varios tipos: get, post, put, delete, etc.
- Permiten enviar parámetros y recibir respuestas HTTP con datos del servidor.
- Pueden trabajar síncrona o asincronamente con el paradigma de observables y/o promises.

Vuex

- Se trata de un módulo de Vue que permite gestionar un almacen de ámbito global a la instancia y sus componentes.
- Consta de:
  - State: contiene los datos del almacen
  - Mutations: funciones para modificar el almacen
  - Getters: funciones para acceder al almacen
  - Acciones: eventos que permiten acceder a un back-end para gestionar/almacenar los datos en un SGBD.

Enrutamiento

- **VueRouter**
  - es un módulo que permite a Vue gestionar la navegación capturando los clicks sobre los enlaces y sin salir en ningún momento de la ejecuciónde Vue → Single page.
- Cada enlace llama a una página que en realidad es un componente dependiente de la instancia Vue.
- Se debe crear una tabla de rutas y cargarla al crear la instancia.
- En lugar de usar <a href=...> se usa <router-link...>
- Y, dentro de la vista, el contenido de cada página a la que queremos navegar se cargará dentro de
  - <router-view>...</router-view>

i18n

- Para internacionalizar y/o localicar nuestra aplicación con Vue, podemos usar el módulo Vue-i18n.
- Se debe declarar un objeto con los textos en cada idioma.
- Antes de declarar la instancia Vue, se declarará un objeto Vuei18n con los textos localizados y éste se le pasará a la instancia en su declaración.
- Este módulo proporciona una función para obtener los textos localizados:

```
{{ $t('nombre.del.recurso') }}
```

i18n

- Ejemplo:

```
const textos= {
   es: { home: { saludo: 'Hola!' } }
}
const i18n = new Vuei18n({
   locale: 'es',
   messages: textos
});
var app = new Vue({
    el: ‘#app’,
    i18n,
    ...
})
```