Desarrollo de Aplicaciones Web

Desarrollo web en entorno cliente

Tema 8: Introducción a Vue

1. Vue

Vue es una herramienta JavaScript para la construcción de interfaces de usuario y aplicaciones desde el lado del cliente.

Su creador, Evan You, lo define como un framework progresivo, ya que está dividido en librerías bien acotadas, cada una con un cometido específico. Su núcleo es pequeño, enfocado sólo en la capa de visualización (como React) pero diseñado para crecer, ya que es fácil añadirle bibliotecas o proyectos existentes, algunos desarrollados por el mismo equipo de Vue. De esta manera, el desarrollador va incluyendo los diferentes módulos según las necesidades del contexto en el que se encuentre. No es necesario incluir toda la funcionalidad desde el principio como en el caso de otros frameworks como AngularJS 1.x.

Vue tiene una curva de aprendizaje menor que otros frameworks y es extremadamente rápido y ligero. Además, presume de ser una librería bastante rápida que consigue renderizar en mejores tiempos que ReactJS.

Por lo tanto, de forma resumida:

- Se trata de un framework progresivo para construir interfaces de usuario.
- Diseñado para ser adoptado gradualmente.
- Permite crear SPAs.

2. Primera aplicación con Vue

Para empezar a hacer uso de Vue, es necesario incluir la dependencia de Vue de alguna manera. Una forma muy sencilla de incorporar Vue a los proyectos web es a través de un CDN. Otras formas son: descargándolo, instalarlo a través de npm, haciendo uso de Vue CLI, etc.

Para enlazarlo desde un CDN, con Vue 3:

```
<script src="https://unpkg.com/vue@next"></script>
```

A través de un CDN es buena forma de comenzar con Vue, pero no es la forma más recomendable de trabajar cuando ya se tienen ciertos conocimientos del framework.

Modelo Vista Controlador

Toda aplicación Vue debería contener los siguientes componentes básicos:

- El modelo: Son los datos de la aplicación. Se trata de un objeto JavaScript con propiedades y sus valores iniciales (el objeto data que se verá más adelante).
- La vista: Será la plantilla HTML.
- El controlador: Una instancia Vue que enlaza el modelo con la vista permitiendo que se comuniquen entre ellos.

La idea es que el modelo y la vista permanezcan sincronizados, de tal forma que un cambio en el modelo haga que se actualice de forma automática la vista, y viceversa.

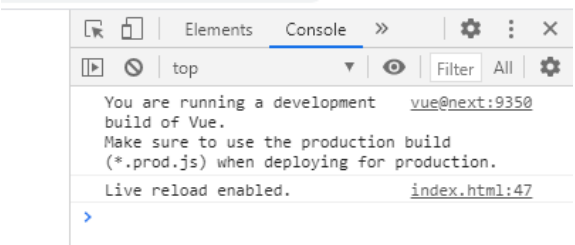
Documento html

Para crear la primera aplicación con Vue, vamos a seguir los siguientes pasos básicos:

- En el documento html creado, se enlazará la librería de Vue en la sección head.
- Vue se ejecutará dentro de un elemento de la página, al que se suele asignar como id el valor “app” (aunque puede ser cualquier otro). Por ejemplo, un elemento de tipo div.
- Dentro del elemento nombrado en el punto anterior, se pueden usar expresiones Vue.

El elemento dentro del cual se va a ejecutar Vue, no tiene por qué ser el único elemento de la página. Fuera de él se puede incluir HTML “normal”, que se ejecutará como ya conocemos, además de poder haber más de un elemento en el que se ejecute Vue.

Si se ejecuta el proyecto con Vue ya enlazado, en la consola deberá aparecer un mensaje como el siguiente:



Documento JavaScript

En el fichero JavaScript del proyecto hay que crear un nuevo objeto Vue, que se almacenará en una constante.

En Vue 3, la sintaxis es diferente a la de versiones anteriores:

- la instancia del objeto Vue se crea con el método createApp, no con el constructor de Vue.
- la propiedad data será una función que devuelve un objeto. Dicho objeto es el que define e inicializa todas las variables que se van a usar en la vista.
- el elemento en que se montará la aplicación no se incluye en el objeto que se pasa. Es necesario montar la aplicación con el método mount de la instancia creada, al que se pasará como parámetro el selector CSS del elemento en el que se va a ejecutar Vue.

```
const app = Vue.createApp({
    data() {
      return { … }
    }
  });
```

Expresiones Vue en elementos html

Dentro del elemento al que se ha asignado un id con valor “app”, es posible utilizar expresiones (o directivas) de Vue.

El uso de *{{ expresión }}* permite escribir expresiones de JavaScript, es decir, posibilita ejecutar Javascript dentro del HTML. Es una representación de datos al DOM de forma más sencilla. Esta es una característica de este tipo de frameworks (es igual para React o Angular) por la que enlazan el DOM y los datos de forma que cualquier cambio en uno se refleja automáticamente en el otro.

Este tipo de enlace unidireccional se denomina interpolación, es una forma de binding, y viene representado por las ya mencionadas dobles llaves.

EJERCICIO 1

- Descarga el proyecto “Primeros pasos con Vue” que encontrarás en el Aula Virtual de la asignatura.
- Al abrir la página web en un navegador aparece el mensaje ¡Hola Mundo!. Cambia ese mensaje por: “¿Esto es Vue?”

3. Directivas

Las directivas son atributos personalizados que se ponen en los elementos HTML y que permiten extender el comportamiento por defecto de los mismos. Todas comienzan por **v-**. Las más comunes son:

- **v-text**: es equivalente a hacer una interpolación.
- **v-once**: igual que la anterior, pero sin cambiar el texto aunque cambie el valor de la variable que se muestra.
- **v-html**: permite que el texto que se muestra contenga caracteres HTML que interpretará el navegador (con interpolación las etiquetas HTML son escapadas).
- **v-bind**: para asignar el valor de una variable a un atributo de una etiqueta HTML.
- **v-model**: permite enlazar un input a una variable.
- **v-if**: renderiza o no el elemento que la contiene en función de una condición.
- **v-show**: similar al v-if pero siempre renderiza el elemento (está en el DOM) y lo que hace es mostrarlo u ocultarlo (display: none) en función de la condición. Es mejor si el elemento va a mostrarse y ocultarse a menudo porque no tiene que volver a renderizarlo cada vez.
- **v-for**: repite el elemento HTML que contiene esta etiqueta para cada posición de un array.
- **v-on**: le pone al elemento HTML un escuchador de eventos (ej <button v-on:click="pulsado">Pulsa</button>.

3.1. v-bind

En el punto 2 se ha visto una primera forma de binding, interpolación, pero no es la única que Vue ofrece. Con v-bind es posible enlazar un dato como atributo de un elemento html.

```
<div id="app">
        <p v-bind:title="mensaje"> Sitúa el ratón encima de este texto </p>
</div>
```

En el caso concreto de v-bind, ya que es muy usada, puede omitirse la primera parte. En el caso del ejemplo anterior, sería lo mismo poner:

```
<p :title="mensaje"> Sitúa el ratón encima de este texto </p>
```

3.2. v-model

Como tercera forma de binding, encontramos el enlace bidireccional. Con la directiva v-model es posible enlazar un campo de un formulario con una variable. Así, si se introduce un nuevo valor (campo de tipo input), si se selecciona una opción diferente a la actual (select), etc. se pueden actualizar datos con los nuevos valores.

La actualización es automática. Siguiendo con el ejemplo del punto anterior, al introducir valor en el input se actualiza de forma automática el valor mostrado al pasar el ratón sobre el div:

```
<input v-model="mensaje">
```

3.3. v-if / v-else

- v-if permite renderizar, o no, un elemento en función de una variable o expresión.
- La directiva v-else, siendo opcional, de incluirse deberá ir inmediatamente después del elemento que incluya v-if.
- Existe una tercera opción, v-else-if, que permite enlazar más de dos opciones.

```
<h2 v-if="tipo === 1"> A </h2>
<h3 v-else-if="tipo === 2"> B </h3>
<h4 v-else-if="tipo === 3"> C </h4>
<h1 v-else> No es ni 1, ni 2, ni 3. </h1>
```

3.4. v-for

Haciendo uso de esta directiva es posible mostrar una lista de elementos HTML a partir de los datos de un array concreto. La directiva v-for recorrerá el array y, para cada posición, ejecutará la acción indicada.

El valor de esta directiva se puede representar de varias maneras:

- recogiendo un ítem del array cada vez: v-for="libro in libros"
- recuperando para cada posición del array, además del valor, el índice de la misma: v-for="(libro,index) in libros"
- ejecutándolo sobre un rango: v-for="n in 10"

Vue renderiza más eficientemente si cada elemento que crea v-for tiene su propia clave, lo que se consigue con el atributo key. La clave será un campo único del elemento o el índice:

```
:key="index" ó v-bind:key="index"
```

** No se recomienda usar v-for y v-if en el mismo elemento. Siempre se ejecuta primero el v-if.

EJERCICIO 2

- En el proyecto “Fundamentos básicos de Vue” encontrarás ejemplos de lo visto hasta el momento. Ejecútalo e inspecciona, a través de la consola, los elementos que se han declarado. Plantea las dudas sobre directivas que te surjan.
- Crea una página web que contenga dos elementos div paralelos:
  - En el primer div, dependiendo del valor de una propiedad “elemento”, se mostrará:
    - Cuando el valor sea “input”: un input (no hace falta formulario).
    - Cuando el valor sea “texto”: un span con el texto incluido en la propiedad texto (el texto será “Esto es un span”).
    - Cuando el valor sea “tabla”: una tabla con una fila y dos columnas.
  - En el segundo contenedor div, a partir de un array con datos de personas, se creará una tabla con tres columnas que solo se mostrará si hay algún dato que mostrar:
    - En la primera columna se mostrará el dni de cada persona.
    - En la segunda el nombre.
    - En la tercera los apellidos.
  - Deberás crear los objetos Vue necesarios con toda la información que se pide.

3.5. v-on Eventos

Con la directiva v-on, es posible capturar un evento e indicar la ejecución de un método como respuesta al mismo.

```
<button v-on:click="mostrarAlerta"> Mostrar alerta</button>
```

Al igual que pasa con v-bind, se trata de una directiva muy usada por lo que también se puede abreviar. En este caso, bastaría con poner una @ delante del evento a capturar:

```
<button @click="mostrarAlerta"> Mostrar alerta</button>
```

El método a ejecutar, hay que incluirlo en el objeto Vue. Para ello, se añade una nueva propiedad llamada methods en el objeto, a continuación de data(). Y dentro de dicho elemento, se incluirán todos los métodos necesarios.

Desde los métodos incluidos en el elemento methods, es posible acceder a los datos que se han incluido en data() a través de **this**.

```
data() {
        return {
                mensaje: "Hola Mundo"
        };
},
methods: {
        metodo1() {
                alert(this.mensaje);
        }
}
```

Es posible pasar parámetros a la función manejadora del evento desde el documento HTML.

Modificadores de eventos

Utilizando el . como separador, se puede añadir un modificador a un evento capturado con la directiva v-on. Los modificadores más comunes son:

- .prevent: equivale a hacer un preventDefault().
- .stop: misma funcionalidad que stopPropagation().
- .self: para evitar el evento en caso de producirse en algún hijo del elemento con la directiva, es decir, sólo se lanza si el evento se produce en dicho elemento.
- .once: sólo se lanza la primera vez que se produce el evento.

```
<form @submit.prevent="enviarFormulario">
```

EJERCICIO 3

Crea una página web para poder crear una lista de propósitos de año nuevo. Debe cumplir con los siguientes requisitos:

- Habrá un título inicial con el texto “Propósitos de año nuevo: “

- Se incluirá una lista no numerada de propósitos de año nuevo que solo se mostrará si hay al menos un propósito registrado. En caso contrario se mostrará el texto “La lista de propósitos está vacía”
- Tendrá un checkbox para cada propósito (se podrá marcar/desmarcar para cambiar su estado)
- El texto del propósito aparecerá tachado si ya se ha llevado a cabo (si está seleccionado el check). **Mira el elemento HMTL *del*
- Al hacer doble click en un propósito debe borrarse de la lista.
- Habrá, también, un input con un botón para añadir un nuevo propósito a la lista. Sólo se añade si se ha introducido texto.
- Por último, se incluirá un botón que borrará toda la lista de propósitos tras pedir confirmación al usuario.

**A tener en cuenta:

- El elemento data del objeto Vue será algo similar a:

```
data() {
        return {
                nuevoProp: "",
                propositos: [{
                        texto: "Hacer deporte",
                        hecho: false
                        }, {
                        texto: "Comer mas sano",
                        hecho: false
                        }, {
                        texto: "Viajar mas",
                        hecho: true
                }]
        };
}
```

- Por cada propósito habrá elemento de tipo checkbox y un texto.
- Cada checkbox estará enlazado con su propiedad “hecho”.
- el input se enlazará con una variable, en el ejemplo anterior llamada nuevoProp. De ahí se sacará el valor para incluirlo en el array de propósitos cuando sea necesario.

4. Clases dinámicas y propiedades computadas

** Los ejemplos de este apartado se encuentran en el proyecto “Clases dinámicas y Propiedades computadas”. Lo encontrarás en el Aula Virtual de la asignatura.

Métodos

Hasta el momento se ha visto el uso de métodos. Se trata de funciones, incluidas en el apartado methods del objeto Vue, que tiene diferentes propósitos.

Estas funciones se comportan de igual forma que se ha visto para JavaScript puro, es decir, se pueden llamar desde cualquier parte y evitar la repetición de código, se pueden utilizar como manejadoras de eventos, se crean para incluir funcionalidad en la aplicación, etc.

Para acceder a una propiedad del data desde un método o para hacer una llamada a otro método, es necesario hacerlo a través de *this*.

Asignación de clases dinámicas

Haciendo uso de la directiva v-bind, es posible asignar clases a los diferentes elementos HTML de forma dinámica. Algunas de las opciones para hacerlo son:

- Asignar a una variable, por código, las distintas clases separadas por un espacio en blanco y, en el elemento HTML, bindear dicha variable.
- Hacer bind de un objeto donde cada propiedad es el nombre de una posible clase y su valor es un booleano que indica si tendrá o no dicha clase.
- Indicar las clases en forma de array de variables, donde el valor de cada variable será el nombre de una clase.
- Asignar clases dependiendo de alguna validación.

Propiedades computadas

También llamadas calculadas, se trata de variables cuyo valor hay que calcularlo haciendo uso de alguna función.

Este tipo de propiedades cobran mas sentido cuando se hace uso de componentes, como se verá en temas posteriores. No se recomienda colocar demasiada lógica en las plantillas HTML para no complicar la interpretación de los componentes, por eso, es recomendable sacar esos cálculos a una propiedad computada.

Toda función añadida en la sección computed debe devolver algún valor. Las principales diferencias entre métodos y propiedades computadas son:

- las segundas se almacenan en caché según sus dependencias, y solo se reevalúan cuando alguna de éstas cambia. Por su parte, un método se ejecutará cada vez que se renderice, aunque no vaya a producirse un resultado diferente.
- las propiedades calculadas son reactivas, por lo que su valor se actualiza solo, mientras que los métodos no reaccionan a cambios en las propiedades del data que estén tratando.

No es recomendable llamar a métodos desde una propiedad calculada.

Estos métodos que van a realizar los cálculos, no se incluyen en el apartado methods del objeto Vue, sino que se añaden en un elemento nuevo, **computed**.

```
app = Vue.createApp({
  data() {
    return {
      ...
    };
  },
  methods: {
    metodo1() { ...},
    …
  },
  computed: {
    metodoComp1() { … },
    …
  }
});
```

EJERCICIO 4

Siguiendo con lo implementado en el ejercicio 3:

- En el fichero de estilos estarán definidos los siguientes estilos, asignados por clase:
  - Con color verde, texto alineado a la derecha y fuente “Gill Sans”.
  - Con color rojo, fuente “fantasy” y tipo de icono de lista “square”.
  - Con color amarillo, color de fondo negro y texto centrado.
  - Con color rosa, color de fondo vende oscuro y texto centrado.
- Al título inicial se le aplicarán los siguientes estilos:
  - Cuando haya más de 3 propósitos cumplidos en la lista, los estilos indicados en el tercer punto.
  - Cuando hay mas de 5 propósitos cumplidos en la lista, los estilos indicados en el cuarto punto.
  - En cualquier otro caso, ningún estilo.
- A cada uno de los propósitos, se aplicarán:
  - Si está cumplido, los estilos del primer punto.
  - En caso contrario, los estilos del segundo punto.

5. Herramientas para trabajar con Vue

5.1. Vue devtools

Extensión tanto para distintos navegadores que permite inspeccionar el objeto Vue y acceder a todos los datos. De gran ayuda especialmente si se hace uso de componentes.

En la herramienta de desarrollador aparecerá una nueva opción, Vue, con 4 botones:

- Componentes: Vista por defecto. Permite inspeccionar los componentes Vue.
- Vuex: es la herramienta de gestión de estado para aplicaciones medias/grandes.
- Eventos: permite ver todos los eventos emitidos.
- Refrescar: refresca la herramienta.

Junto al componente puede aparecer "**= $vm0**". Esto indica que DevTools ha creado una variable con ese nombre que contiene el componente para inspeccionarlo desde la consola.

Al inspeccionar componentes, bajo la barra de botones aparecerá otra barra con 3 herramientas:

- Buscar: permite buscar el componente con el nombre introducido.
- Seleccionar componente en la página: al pulsarlo (se dibuja un punto en su interior) hace que al pulsar sobre un componente en la página se seleccione en la herramienta de inspeccionar componentes.
- Formatear nombre de componentes: muestra los nombres de componentes en el modo camelCase o kebab-case.

**Si se quisiera trabajar sin servidor web (desde file://…), habría que habilitar el acceso a ficheros en la extensión.

5.2. Extensiones para el editor de código

Al trabajar con componentes, se hará uso de ficheros con extensión **.vue**, que integran el HTML, el JS y el CSS de cada componente. Para que el editor los detecte correctamente es conveniente instalar la extensión para Vue (por ejemplo, **Vetur** o **es6-string-html**).

También pueden ser de gran ayuda **Syntax Highlight for Vue.js** o **Vue 3 Snippets**.