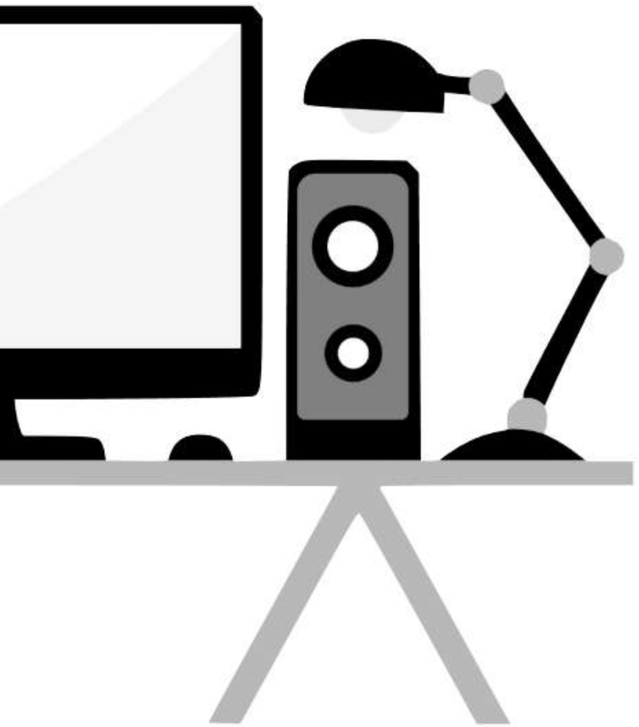
MANEJO DE EVENTOS EN VUE

Lección 04

Competencia General

Reconocer los elementos fundamentales del Domain Object Model y los mecanismos para la manipulación de elementos en un documento HTML.

Contenidos



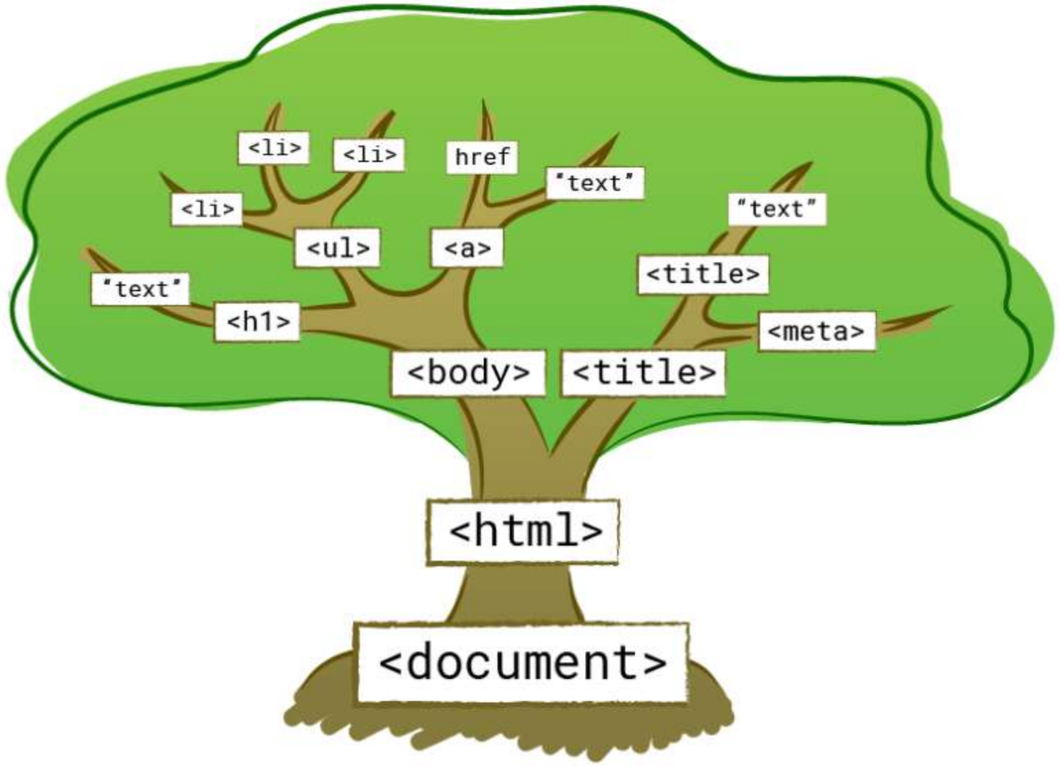
1. Recordatorio DOM
2. Qué es un evento en VUE
3. Principales eventos que se pueden manejar en el DOM
4. Métodos manejadores de eventos
5. Métodos manejadores en línea
6. Modificadores de eventos
7. Modificadores de teclas

Qué es el DOM

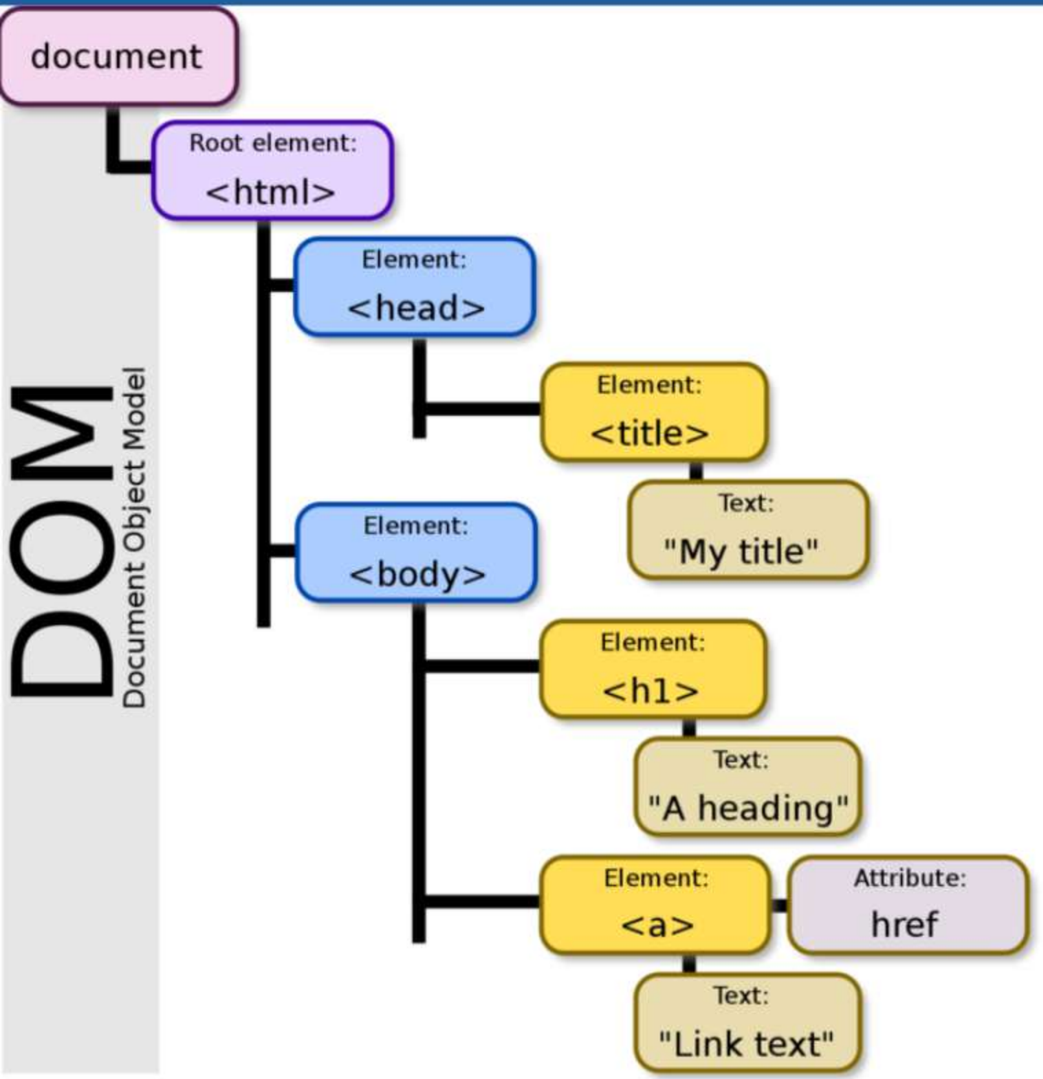
Recordando

El DOM (Document Object Model) es una jerarquía de objetos del navegador y se encuentra definido y administrado por la W3C.

Cada objeto en la jerarquía del DOM modelan tanto la ventana del navegador como el historial y los elementos de la página como párrafos, tablas, formularios, etc.



Jerarquía de elementos del DOM



Recordando

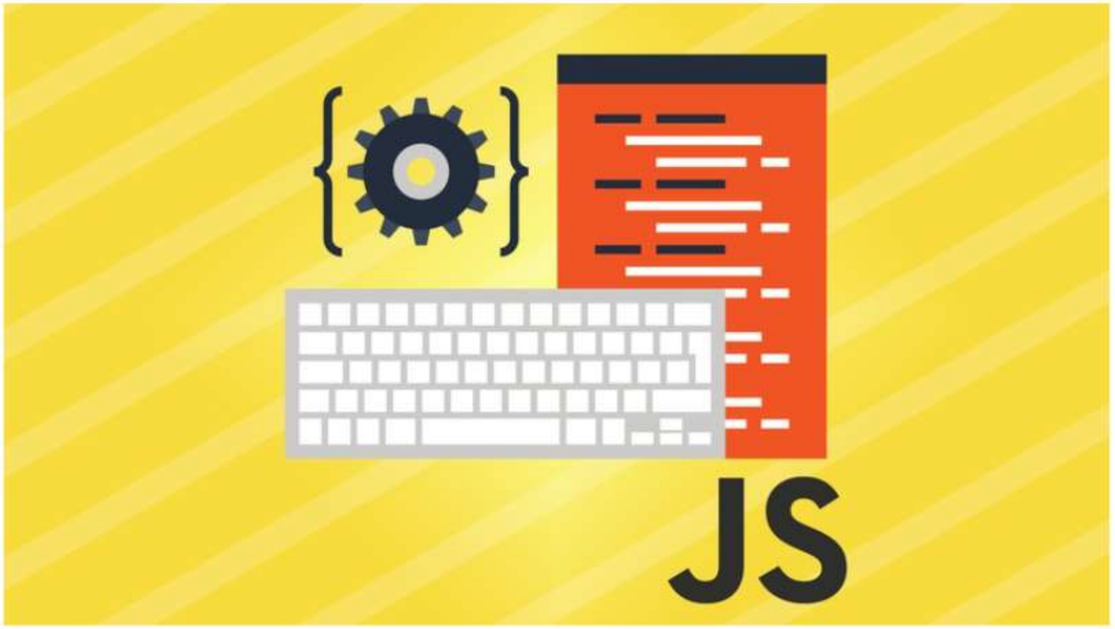
El objeto superior padre de todos es **window**, de el derivan objetos con funcionalidades distintas entre esos objetos se encuentra **document** que contiene cada elemento del html con sus atributos para desplegarse en pantalla.

Qué es un evento

Recordando

En JavaScript, la interacción con el usuario se consigue mediante la captura de eventos. Un evento es una acción del usuario ante la cual puede realizarse algún proceso.
Por ejemplo validar un campo de un formulario.

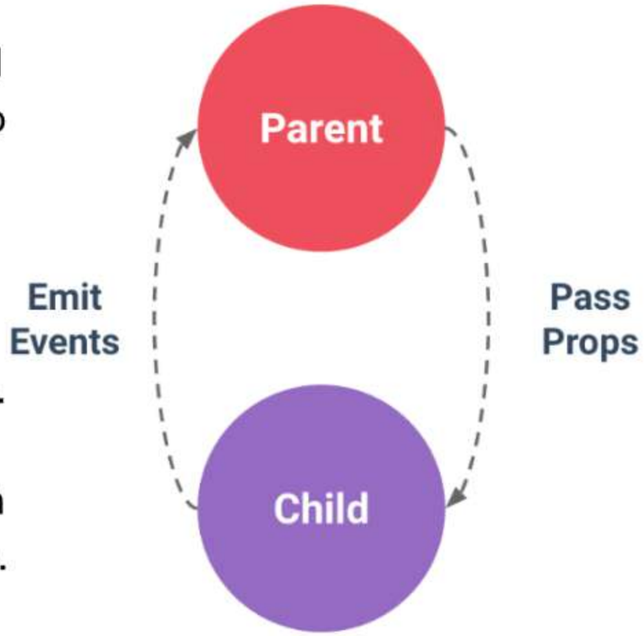
En **Vue**, el manejo de eventos DOM se realiza de manera más sencilla mediante la directiva v-on, o usando su abreviación @.



¿Qué es un evento en Vue?

En Vue, un evento representa una acción del usuario que el sistema puede detectar y manejar. A diferencia del DOM tradicional, Vue permite gestionarlos de forma declarativa, clara y reactiva directamente desde el template del componente.

Los eventos en Vue simplifican la lógica de interacción al permitir ejecutar métodos ante acciones del usuario sin necesidad de escribir código adicional en JavaScript puro para seleccionar elementos o agregar listeners.



Eventos y componentes

En el contexto de componentes, los eventos son la contraparte de las props:

- **Las props** permiten pasar datos del **componente padre al hijo**.
- **Los eventos** permiten enviar información desde el **hijo hacia el padre**, notificando que ocurrió un suceso (por ejemplo, un clic, un envío de formulario o una acción del usuario).

Ejemplo típico:

Un botón dentro de un componente hijo puede emitir un evento para que el componente padre realice una acción como abrir un modal, mostrar un mensaje o actualizar un dato.

Principales eventos que se pueden manejar en el DOM

Los eventos DOM se manejan de forma declarativa utilizando la directiva **v-on** o su forma abreviada @. Esto permite reaccionar fácilmente ante acciones del usuario, como clics, movimientos del mouse, entradas de teclado, entre otros, sin necesidad de manipular directamente el DOM con addEventListener().

Evento	Descripción
click	Cuando el usuario hace clic sobre un elemento
dblclick	Doble clic
mouseenter	Cuando el mouse entra en un elemento
mouseleave	Cuando el mouse sale de un elemento
mousedown	Cuando se presiona un botón del mouse
mouseup	Cuando se suelta el botón del mouse
keydown	Cuando se presiona una tecla
input	Cuando se ingresa o cambia texto en un input o textarea
submit	Cuando se envía un formulario
change	Cuando cambia el valor de un input, checkbox o select

sustantiva
CON•SENTIDO

Limitaciones del manejo de eventos en HTML nativo

Cuando trabajamos con HTML y JavaScript puro, una de las formas más comunes de manejar eventos como los clics es utilizando atributos como onClick directamente en el HTML.

Veamos el siguiente ejemplo:

```
<button onClick="showMessage()">Click</button>

<script>
  const showMessage = () => alert("¡Hola! ¡Gracias por hacer clic!");
</script>
```

Este enfoque muestra un mensaje al hacer clic en el botón, pero presenta varias desventajas: acopla directamente el HTML con el JavaScript, exige que la función esté en el ámbito global y no permite separar la lógica de presentación de la de comportamiento, lo que dificulta la reutilización y escalabilidad del componente.

Limitaciones del manejo de eventos en HTML nativo

En Vue, se pueden manejar eventos del DOM de forma simple y declarativa usando la directiva v-on o su versión abreviada @.

Esto permite separar la lógica del comportamiento y la presentación, manteniendo el código más limpio y reutilizable.

Además, Vue permite definir eventos **de forma dinámica** usando la sintaxis @[evento]. Esto es útil cuando el tipo de evento a manejar puede cambiar en tiempo de ejecución.

Por ejemplo, si la variable evento contiene el valor "click", Vue interpretará la instrucción como @click="mostrarDinamico".

```
<div id="app">
  <button v-on:click="mostrarMensaje">Usando v-on</button>
  <button @click="mostrarAbreviado">Usando @</button>
  <button @[evento]="mostrarDinamico">Evento dinámico</button>
</div>

<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
<script>
const { createApp, ref } = Vue

createApp({
  setup() {
    const evento = ref('click')

    const mostrarMensaje = () => alert('Con v-on')
    const mostrarAbreviado = () => alert('Con @')
    const mostrarDinamico = () => alert(`Usando evento dinámico: ${evento.value}`)

    return { evento, mostrarMensaje, mostrarAbreviado, mostrarDinamico }
  }
}).mount('#app')
</script>
```

Métodos manejadores de eventos

Un **método manejador de eventos** es una función definida en la sección methods de un componente. Su propósito es **responder a eventos del DOM**, como clics, pulsaciones de teclas o cambios en formularios.

Estos métodos se vinculan a los eventos mediante la directiva v-on o su forma abreviada @.

En este ejemplo, se define un botón utilizando la directiva @click, que permite vincular el evento del DOM directamente con una función o método del componente.
Cuando el usuario hace clic en el botón, se ejecuta la función mostrarAlerta, la cual está definida dentro de la sección methods del componente. Esta función muestra una alerta en pantalla como respuesta al evento detectado.

```
<template>
  <button @click="mostrarAlerta">Haz clic aquí</button>
</template>

<script>
export default {
  methods: {
    mostrarAlerta() {
      alert('¡Evento manejado con éxito!');
    }
  }
}
</script>
```

Métodos manejadores en línea

Además de vincular eventos a funciones definidas en el componente, también es posible definir métodos directamente en el template, de forma **en línea**.

Esto se conoce como **métodos manejadores en línea**, y permite ejecutar expresiones simples directamente desde el HTML sin necesidad de declararlas en methods o setup(). Es útil para casos rápidos o cuando no se requiere lógica compleja.

En este ejemplo, el botón tiene un manejador en línea: @click="contador++". Esto incrementa la variable contador directamente cada vez que el usuario hace clic.

No fue necesario declarar una función para esta acción, ya que la operación es sencilla y puede expresarse en una sola línea.

```
<div id="app">
  <button @click="contador++">Incrementar</button>
  <p>Contador: {{ contador }}</p>
</div>

<script
src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
<script>
const { createApp, ref } = Vue

createApp({
  setup() {
    const contador = ref(0)
    return { contador }
  }
}).mount('#app')
</script>
```

Modificadores de eventos

Los modificadores de eventos son sufijos especiales que se agregan a la directiva v-on (o su versión abreviada @) usando un punto (.).
Su propósito es **alterar el comportamiento predeterminado del evento**, de forma declarativa y sencilla, sin necesidad de escribir código extra en el manejador.

Modificador	Descripción
`.prevent`	Llama a `event.preventDefault()`. Evita el comportamiento por defecto del evento, por ejemplo, que un formulario se envíe al hacer clic en un botón.
`.stop`	Llama a `event.stopPropagation()`. Detiene la propagación del evento hacia elementos padre, útil cuando no quieres que múltiples manejadores se activen por un mismo evento.
`.once`	Ejecuta el manejador de evento solo una vez.
`.capture`	Escucha el evento durante la fase de captura en lugar de la fase de burbuja. Se usa en casos avanzados donde se requiere interceptar el evento antes que otros.
`.self`	El manejador solo se ejecuta si el evento se origina directamente en el elemento, no en un hijo anidado.
`.passive`	Indica que el manejador nunca llamará a `preventDefault()`, mejorando el rendimiento en ciertos eventos como `touchstart` en dispositivos móviles.

`.left`
`.right`
`.middle`
	Se activa solo si el evento es generado con el botón izquierdo/derecho/central del mouse.

Modificadores de teclas

Además de los modificadores comunes como los mostrados anteriormente, Vue permite utilizar **modificadores de teclas** para ejecutar métodos sólo cuando el usuario presiona teclas específicas durante un evento de teclado.

Por ejemplo, puedes usar .enter en un evento @keydown para que el método se ejecute solo si el usuario presiona la tecla Enter:

```
<input @keydown.enter="enviarFormulario" placeholder="Presiona Enter">
```

De forma similar, puedes usar .esc para que una acción se ejecute solo cuando el usuario presione la tecla Escape:

```
<input @keydown.esc="cancelarAccion" placeholder="Presiona Escape">
```

Este tipo de modificadores permiten controlar con mayor precisión la interacción del usuario con el teclado, sin necesidad de escribir lógica adicional como event.key === 'Enter'.

Modificadores de teclas

Además de .enter y .esc, Vue ofrece una variedad de **modificadores de teclas específicos** que permiten ejecutar métodos sólo cuando el usuario presiona ciertas teclas.

A continuación, se muestran algunos de los modificadores de teclas más conocidos:

Modificador	Descripción
.enter	Cuando se presiona la tecla Enter
.tap	Cuando se presiona la tecla Tab
.delete	Se activa con Delete o Backspace
.esc	Cuando se presiona la tecla Escape
.space	Cuando se presiona la tecla Espacio
.up	Cuando se presiona la flecha arriba
.down	Cuando se presiona la flecha abajo
.left	Cuando se presiona la flecha izquierda
.right	Cuando se presiona la flecha derecha

sustantiva

CON•SENTIDO