



JavaScript

Control de Eventos 3

< Ing := Carlos H. Rueda C++ />

* * * * *

Eventos personalizados

Eventos Burbujeantes (bubbles)

El evento burbujeante es un concepto en el DOM (Modelo de Objeto del Documento) de JavaScript.

Sucede cuando un elemento recibe un evento, y ese evento burbujea (o puedes decir que es transmitido o propagado) a sus elementos, padres y ancestros en el árbol del DOM hasta que llega al elemento raíz.

Eventos Burbujeantes (bubbles)

En otras palabras, cuando un evento se dispara en un elemento, el evento se propaga a través de los elementos secundarios y ascendentes del elemento original.

Esto significa que si un elemento secundario tiene un controlador de eventos, el controlador de eventos del elemento secundario se activará antes que el controlador de eventos del elemento ascendente.

* * * * *

Ejemplo

```
<style>
  body {
    background-color: wheat;
  }
  div {
    background-color: burlywood;
    width: 40%;
    height: 100px;
    text-align: center;
  }
</style>
```

Ejemplo

```
<style>
  span {
    background-color: chocolate;
    padding: 20px;
    margin: 10px auto;
    width: 100px;
    display: block;
    text-align: center;
  }
  i {
    background-color: coral;
    display: inline-block;
    padding: 5px;
    text-align: left;
  }
</style>
```

Ejemplo

```
<body>
  <div>
    <span>
      <button>Click me!</button>
    </span>
    <i></i>
  </div>
```

* * * * *

Ejemplo

```
<script>
  document.querySelector("button").addEventListener("click",
    e => document.querySelector("i").textContent =
      `Me cliqueaste en ${e.target.nodeName}`);

  document.querySelector("span").addEventListener("click",
    e => alert(`Me cliqueaste en SPAN`));

  document.querySelector("div").addEventListener("click",
    e => alert(`Me cliqueaste en DIV`));

  document.body.addEventListener("click",
    e => alert(`Me cliqueaste en BODY`));
</script>
</body>
```

Custom Events

Crear un **evento personalizado** en JavaScript implica **instanciar el objeto CustomEvent**, al que se le proporciona un nombre para el evento como primer parámetro.

También se puede utilizar el **objeto Event**. Si se desea añadir datos personalizados, se proporciona un objeto de opciones como segundo parámetro. Un ejemplo de definición de un evento personalizado es:



* * * * *

Sintaxis

```
const messageEvent = new CustomEvent("message", options);
```

Esta instancia permite la creación de eventos personalizados con la posibilidad de agregar datos específicos según las opciones proporcionadas.

* * * * *

Nombre del evento personalizado

Cuando se elige un nombre para un evento en JavaScript, es recomendable seguir prácticas como **usar minúsculas, evitar camelCase y optar por kebab-case o namespaces** para hacerlo claro y comprensible. Por ejemplo, **"user:data-message"** indica un evento de usuario que recibe un mensaje de datos. Estas convenciones facilitan la comprensión y la coherencia a largo plazo.

* * * * *

Opciones del evento

Al crear un evento personalizado con **CustomEvent**, el segundo parámetro es un **objeto de opciones** que permite especificar detalles sobre el comportamiento y contenido del evento.

* * * * *



Opciones del evento

Opciones	Valor inicial	Descripción
detail	null	Objeto que contiene la información a transmitir.
bubbles	false	Indica si el evento debe burbujear
composed	false	Indica si la propagación puede atravesar el DOM o no.
cancelable	false	Indica si el comportamiento se puede cancelar con <code>.preventDefault()</code> .

👁 En el siguiente fragmento de código vemos como se declara una instancia de **CustomEvent** llamada **user:data-message**, la cuál tiene ciertas opciones definidas, entre las que se encuentran que:

1. El evento debe **burbujear** hacia arriba en el DOM
2. El evento puede atravesar **Shadow DOM** (*útil cuando son **WebComponents***)
3. El evento contiene información en el atributo **detail**

* * * * *

```
const MessageEvent = new CustomEvent("user:data-message", {  
  detail: {  
    from: "Computing",  
    message: "Hello!",  
  },  
  bubbles: true,  
  composed: true,  
});
```

En el objeto de opciones, se observa que hay un objeto "detail" que es definido por el desarrollador, ya que es el diseñador del evento personalizado. Las demás son opciones del evento que se explicarán más adelante.

Event vs CustomEvent

Los eventos nativos del tipo **Event** son utilizados por el navegador para generar eventos asociados a las acciones del usuario. Específicamente, estos eventos pueden ser del tipo **PointerEvent**. Sin embargo, es viable simular (**false**) eventos de navegador.

* * * * *



Event vs CustomEvent

Ejemplo

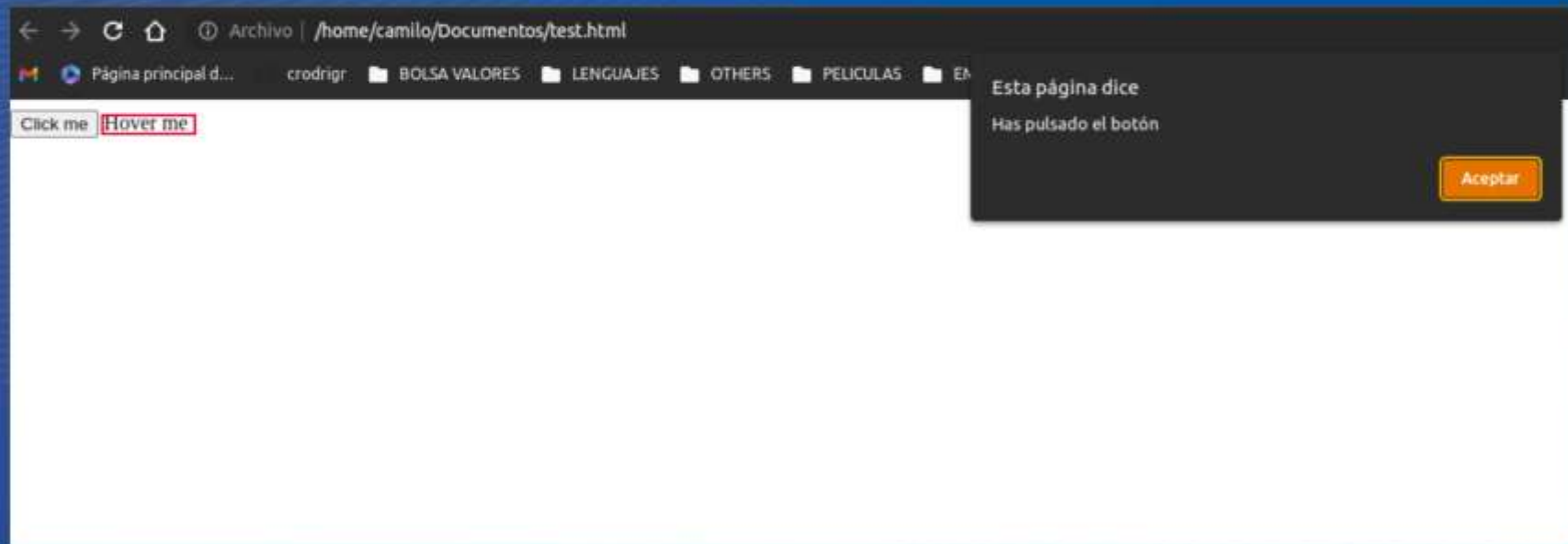
👁 Hemos incorporado un evento que muestra un mensaje al hacer clic en el **button**. Además, hemos agregado otro evento al **span** para que, al mover el ratón sobre él, genere un nuevo evento de clic de ratón y lo envíe al botón. De esta manera, se activará el otro evento en espera, como si el usuario real hubiera hecho clic en el botón. A continuación se observa el código y como se ejecuta en la imagen de abajo.

Ejemplo

```
<button>Clicqueame</button>
<span class="text">Pasa el mouse por encima mio</span>
<script>
  const button = document.querySelector("button");
  const text = document.querySelector(".text");

  button.addEventListener("click", () => alert("Has pulsado el botón"));

  text.addEventListener("mouseenter", () => {
    const event = new Event("click");
    // Lazan manualmente el evento clic del boton
    button.dispatchEvent(event);
  });
</script>
```



En general, los eventos Event se destinan a eventos reales del navegador. Para controlar acciones

específicas, es más común utilizar CustomEvent, al cual se le asigna un nombre y se personaliza según las necesidades.

📢 *“Los CustomEvent, a diferencia de los Event, posibilitan la inclusión de información adicional al crear el objeto, proporcionando mayor flexibilidad en su personalización.”*

Eventos del navegador

Cuando los **usuarios** realizan **acciones**, los **eventos** de navegador se **activan**. Estos eventos nativos son específicos, realizan tareas particulares y proporcionan propiedades o métodos especiales en su objeto de evento para dicha tarea. Aunque generalmente se dispara un evento de tipo **Event**, existen numerosos eventos más específicos en el navegador.



* * * * *

Eventos del navegador

Ejemplo

El siguiente ejemplo utiliza el evento **click** para contar el número de clics consecutivos en cualquier parte de la pantalla, mostrando información como el número de clics (`event.detail`) y las coordenadas (`event.x`, `event.y`) en las que se hizo clic. A continuación se puede ver el código y como se ejecuta.



* * * * *

Eventos del navegador

Ejemplo

```
<style>
  body {
    margin: 0;
    width: 100vw;
    height: 100vh;
    user-select: none;
    font-size: 2rem;
  }
</style>
<body>
  <span></span>
</body>
```

Eventos del navegador

Ejemplo

```
<script>
  const span = document.querySelector("span");
  document.body.addEventListener("click", event => {
    const { x, y, detail } = event;
    span.textContent =
      `Has hecho ${detail} CLICK en las coordenadas (${x}x${y})`;
  });
</script>
```

← → ↻ 🏠 ⓘ Archivo | /home/camilo/Documentos/test.html

🌐 🌐 Página principal d... crodrigr 📁 BOLSA VALORES 📁 LENGUAJES 📁 OTHERS 📁 PELICULAS 📁 EMPRENDIMIENE... 📁 ACADEMIA_ONLINE 📁 IA

Has hecho 1 CLICK en las coordenadas (230x335)

Emisión de eventos

La emisión de eventos en JavaScript, aunque aparentemente sencilla, requiere un entendimiento profundo para evitar sorpresas.

* * * * *



Emisión directa de eventos

Ejemplo

```
<div class="root">  
  <div class="parent">  
    <div class="child">  
      <button>Presióname</button>  
    </div>  
  </div>  
</div>
```

* * * * *

Ejemplo

```
<script>
  const root = document.querySelector(".root");
  const button = document.querySelector("button");

  button.addEventListener("click", () => {
    button.dispatchEvent(new CustomEvent("user:message", {
      detail: {
        name: "Computacion cuantica",
      },
      bubbles: true,
    }));
  });

  root.addEventListener("user:message", event => {
    const name = event.detail.name;
    console.log(`Mensaje recibido de "${name}" (${number})`);
  });
</script>
```

