Somos un ecosistema de desarrolladores de software

</Riwi>

Java Script DOM (Document Object Model)

Eventos

Que es un evento?

En Javascript existe un concepto llamado **evento**, que no es más que una notificación de que alguna **característica interesante** acaba de ocurrir, generalmente relacionada con el **usuario** que navega por la página.
Dichas características pueden ser muy variadas:

- Click de ratón del usuario sobre un elemento de la página
- Pulsación de una tecla específica del teclado
- Reproducción de un archivo de audio/video
- Scroll de ratón sobre un elemento de la página
- El usuario ha activado la opción «Imprimir página»

DOM

El Metodo .appendChild

Forma	Ejemplo	Artículo en profundidad
Mediante **atributos HTML**	<button onClick="..."></button>	Eventos JS desde atributos HTML
Mediante **propiedades Javascript**	.onclick = function() { ... }	Eventos JS desde propiedades Javascript
Mediante addEventListener()	.addEventListener("click", ...)	Eventos JS desde listeners

Evento

propiedades

Saludar es <buttonn>, que esta ubicado en HTML, se dispara el evento click , cuando se haga click, se ejecutara asociado al atributo html, en este caso tenemos un alert( ).

Evento

Organizando la funcionalidad

</Riwi>

```
<script>
  function doTask() {
    alert("Hello!");
  }
</script>

<button onClick="doTask()">Saludar</button>
/// ———————— de forma separada.JS
<script src="tasks.js"></script>
<button onClick="doTask()">Saludar</button>
```

Evento

Eventos mediante Javascript

</Riwi>

```
</Riwi>

<button>Saludar</button>

<script>
const button = document.querySelector("button");
button.onclick = function() {
  alert("Hello!");
}
</script>
```

Evento

Utilizando setAttribute( )

</Riwi>

```
<button>Saludar</button>

<script>
const button = document.querySelector("button");
const doTask = () ⇒ alert("Hello!");
button.setAttribute("onclick", "doTask()");
</script>
```

Evento

El método addEventListener

Eventos Javascript y como gestionarlos a través de código HTML, o a través de código Javascript, utilizando la API del DOM. Sin embargo, la forma más recomendable es hacer uso del método .addEventListener(), el cuál es mucho más potente y versatil para la mayoría de los casos.

Con .addEventListener() se pueden añadir fácilmente varias funcionalidades.
Con .removeEventListener() se puede eliminar una funcionalidad previamente añadida.
Con .addEventListener() se pueden indicar ciertos comportamientos especiales.

Evento

Método .addEventListener( )

Método	Descripción
.addEventListener(STRING event,FUNCTION func)	Escucha el evento event, y si ocurre, ejecuta func.
.addEventListener(STRING event,FUNCTION func,OBJECT options)	Idem, pasándole ciertas opciones.

Evento

Método .addEventListener( )

</Riwi>

```
</Riwi>
```

```
const button = document.querySelector("button");
button.addEventListener("click", function() {
  alert("Hello!");
});
//————————————————————
const button = document.querySelector("button");
function action() {
  alert("Hello!");
};
button.addEventListener("click", action);
//————————————————————
const button = document.querySelector("button");
const action = () ⇒ alert("Hello!");
button.addEventListener("click", action);
```

Evento

Múltiples Listeners (múltiples funciones a un mismo evento)

</Riwi>

```
</Riwi>

<button>Saludar</button>

<style>
  .red { background: red }
</style>

<script>
const button = document.querySelector("button");
const action = () => alert("Hello!");
const toggle = () =>
button.classList.toggle("red");

button.addEventListener("click", action);
// Hello message
button.addEventListener("click", toggle);
// Add/remove red CSS
</script>
```

Evento

Opciones de .addEventListener

Opción	Descripción
BOOLEAN capture	El evento se dispara al inicio (capture), en lugar de al final (bubble).
BOOLEAN once	Sólo ejecuta la función la primera vez. Luego, elimina listener.
BOOLEAN passive	La función nunca llama a .preventDefault() (mejora rendimiento).

Evento

.remove( )

Método	Descripción
.removeEventListener(STRING event,FUNCTION func)	Elimina la funcionalidad func asociada al evento event.

</Riwi>

```
<button>Saludar</button>

<style>
  .red { background: red }
</style>

<script>
const button = document.querySelector("button");
const action = () => alert("Hello!");
const toggle = () =>
button.classList.toggle("red");

button.addEventListener("click", action);
// Add listener
button.addEventListener("click", toggle);
// Toggle red CSS
button.removeEventListener("click", action);
 // Delete listener
</script>
```

Evento

Escuchar eventos y handleEvent

Vamos a trabajar con la clase EventManager, que básicamente gestionará nuestros eventos de una forma más cómoda

</Riwi>

```
class EventManager {
  constructor(element) {
    element.addEventListener("click",
this.sendMessage());  /* Error */
  }

  sendMessage() {
    alert("Has hecho click en el botón");
  }
}

const button = document.querySelector("button");
const eventManager = new EventManager(button);
```

Evento

Mediante funciones (referencia)

</Riwi>

```
class EventManager {
  constructor(element) {
    element.addEventListener("click", this.sendMessage);
  }

  sendMessage() {
    alert("Has hecho click en el botón");
    console.log(this);    // this = referencia al <button>
  }
}
const button = document.querySelector("button");
const eventManager = new EventManager(button);
```

Evento

Escuchar eventos con Objetos

</Riwi>

```
</Riwi>
```

```
const button = document.querySelector("button");
const eventManager = {
  handleEvent: function(ev) {
    alert("¡Has hecho click!");
  }
}
button.addEventListener("click", eventManager);
```

Evento

El objeto Event

</Riwi>

```
const button = document.querySelector("button");
button.addEventListener("click", (event) ⇒ {
  console.log(event);
});
```

Evento

El objeto Event

</Riwi>

```
// Objeto PointerEvent
{
  type: "click",          // Nombre del evento
  pointerType: "mouse"    // Tipo de dispositivo
  altKey: false,          // ¿La tecla ALT estaba presionada?
  ctrlKey: false,         // ¿La tecla CTRL estaba presionada?
  shiftKey: false,        // ¿La tecla SHIFT estaba presionada?
  target: button,         // Referencia al elemento que disparó el evento
  clientX: 43,            // Posición en eje X donde se hizo click
  lientY: 16,            // Posición en eje Y donde se hizo click
  detail: 1,              // Contador de veces que se ha hecho click
  path: [],               // Camino por donde ha pasado el evento
   ...                    // Otros ...
}
```

Evento

Propiedades

Propiedad	Descripción
STRING .type	Indica el tipo de evento en cuestión.
NUMBER .timeStamp	Hora en milisegundos en la que se creó el evento.
BOOLEAN .isTrusted	Indica si es un evento real de un usuario o uno enviado manualmente con .dispatchEvent().

</Riwi>

```
const button = document.querySelector("button");
button.addEventListener("click", (event) => {
  const { type, timeStamp, isTrusted } = event;
  console.log({ type, timeStamp, isTrusted });
}); ...                          // Otros ...
}
```

</Be a coder>