Recuerden poner a grabar la clase

Clase 8

Eventos I

Diplomatura UNTREF

Temario

Eventos:

- ¿Qué es un evento?
- Formas de controlarlo con JS
  - Atributo ON
  - Propiedad ON
  - Metodo AddEventListener
- Manejo del evento On Click
  - Mousemove
  - MouseDown
  - MouseUp

Eventos

¿Que es un evento?

Los eventos en JavaScript son un componente esencial para crear interacciones y dinamismo en aplicaciones web.

Un evento es una acción o suceso que ocurre en una página web, como hacer clic en un botón, mover el ratón, pulsar una tecla, cargar una página o cambiar el tamaño de la ventana del navegador.

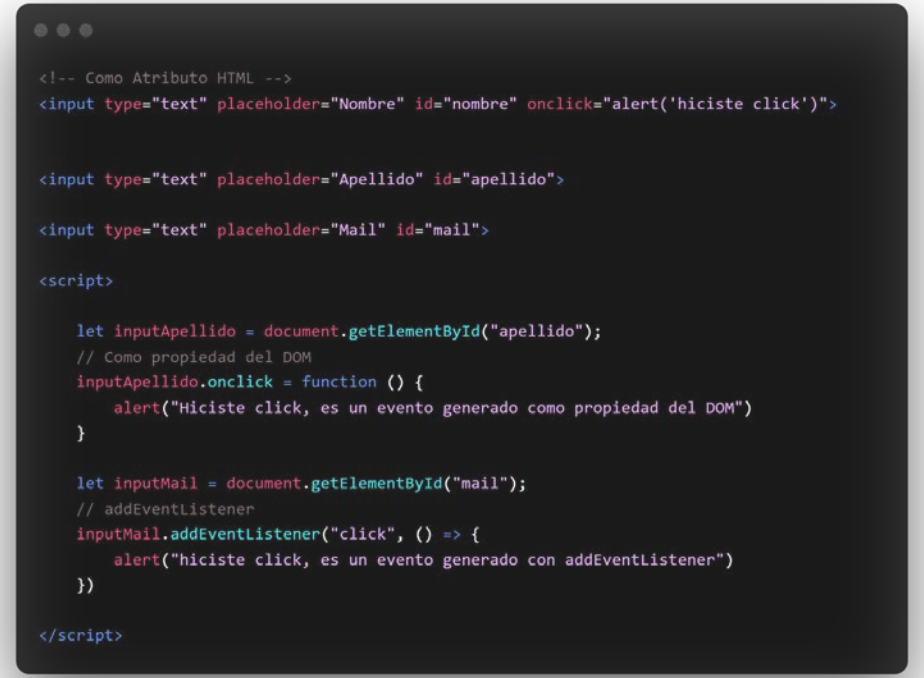
¿Qué es un evento?

Estos eventos permiten que los usuarios interactúen con el contenido de la página y que los desarrolladores respondan a esas interacciones mediante la ejecución de código JavaScript.

Concepto de evento

En JavaScript, los eventos son manejados mediante un patrón de programación conocido como **"Event-Driven Programming"** *(Programación Orientada a Eventos)*.

Este patrón se basa en la idea de que el programa espera y responde a eventos que ocurren en el entorno en lugar de seguir una secuencia de instrucciones lineal.



```
<!-- Como Atributo HTML -->
<input type="text" placeholder="Nombre" id="nombre" onclick="alert('hiciste click')">


<input type="text" placeholder="Apellido" id="apellido">

<input type="text" placeholder="Mail" id="mail">

<script>

    let inputApellido = document.getElementById("apellido");
    // Como propiedad del DOM
    inputApellido.onclick = function () {
        alert("Hiciste click, es un evento generado como propiedad del DOM")
    }

    let inputMail = document.getElementById("mail");
    // addEventListener
    inputMail.addEventListener("click", () => {
        alert("hiciste click, es un evento generado con addEventListener")
    })

</script>
```

Concepto de evento

Cuando ocurre un evento, se genera una notificación, y el navegador o el entorno de ejecución de JavaScript envía ese evento a través del **"Event Loop"** ***(Bucle de Eventos)*** para que pueda ser manejado.

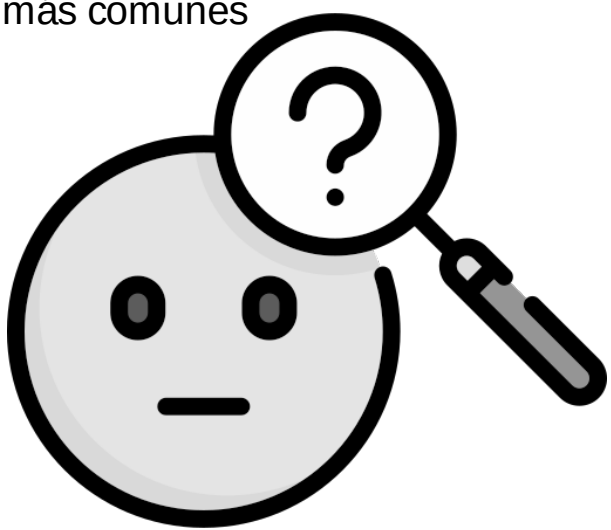
Los desarrolladores pueden registrar "oyentes" (listeners) o "manejadores" (handlers) de eventos que especifican qué acciones deben llevarse a cabo cuando ocurre un evento específico.

```html
<!-- Como Atributo HTML -->
<input type="text" placeholder="Nombre" id="nombre" onclick="alert('hiciste click')">

<input type="text" placeholder="Apellido" id="apellido">

<input type="text" placeholder="Mail" id="mail">

<script>

    let inputApellido = document.getElementById("apellido");
    // Como propiedad del DOM
    inputApellido.onclick = function () {
        alert("Hiciste click, es un evento generado como propiedad del DOM")
    }

    let inputMail = document.getElementById("mail");
    // addEventListener
    inputMail.addEventListener("click", () => {
        alert("hiciste click, es un evento generado con addEventListener")
    })

</script>
```

Tipos de Eventos en Js

Tipos de evento

JavaScript admite una amplia variedad de eventos que pueden ser detectados y manejados, los nombraremos, pero algunos de los eventos más comunes incluyen:



Eventos de ratón: como *"click", "dblclick"* (doble clic), *"mouseover"* (ratón sobre el elemento), *"mouseout"* (ratón fuera del elemento), *"mousemove"* (movimiento del ratón), etc.

Eventos de teclado: como *"keydown"* (tecla presionada), *"keyup"* (tecla liberada) y *"keypress"* (carácter ingresado).

Tipos de evento

Eventos de formulario: como *"submit"* (envío de formulario), "input" (cambio en un campo de entrada), *"focus"* (cuando un elemento recibe foco), **"blur"** (cuando un elemento pierde foco), etc.

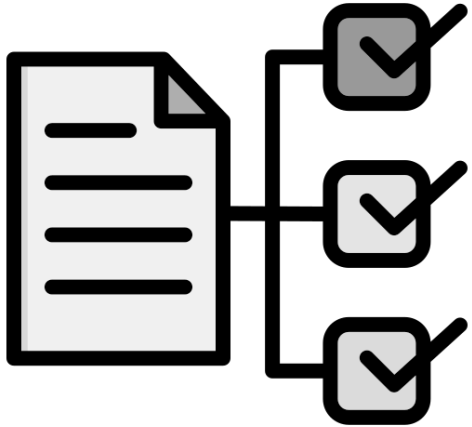
Eventos de ventana: como *"load"* (cuando la página o un recurso se carga completamente), *"resize"* (cuando se cambia el tamaño de la ventana), *"scroll"* (cuando se desplaza la página), etc.

Eventos de tiempo: como *"setTimeout"* (ejecutar una función después de un tiempo determinado) y *"setInterval"* (ejecutar una función periódicamente).

Controlar Eventos en Js

Controlar eventos en Js

Controlar eventos en JavaScript es fundamental para crear interacciones y experiencias dinámicas en aplicaciones web.



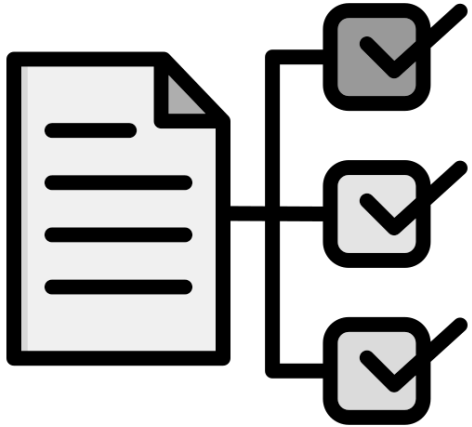
Existen varias formas de controlar eventos en JavaScript: mediante **atributos "on"** en HTML, **propiedades "on"** en JavaScript y el **método "addEventListener"**.

Atributo “on” en HTML

Atributo “on” en HTML

Los **atributos "on"** en HTML son la forma más sencilla y directa de asignar un manejador de eventos a un elemento en el mismo código HTML.

Puedes agregar atributos como **"onclick", "onmouseover", "onkeydown",** etc., directamente en las etiquetas HTML para especificar qué función se debe ejecutar cuando ocurra el evento correspondiente.



Atributo “on” en HTML - en accion

Esta técnica es fácil de implementar y puede ser útil para pequeñas interacciones o prototipos rápidos.

```html
<button onclick="saludar()">Haz clic aquí</button>
```

```
function saludar() {
  console.log("¡Hola, mundo!");
}
```

Atributo “on” en HTML - Atención

Sin embargo, presenta algunas limitaciones importantes:

Mezcla de código: Al utilizar atributos "on" en HTML, se mezcla el código JavaScript directamente con el código HTML, lo que puede dificultar la legibilidad y el mantenimiento del código, especialmente en proyectos más grandes.

Gestión de múltiples eventos: Si necesitas asignar varios manejadores de eventos al mismo elemento, la notación con atributos "on" puede volverse incómoda y poco práctica.

Atributo “on” en HTML - Atención

Dificultad para desvincular eventos: No es fácil desvincular un evento asignado con atributos "on" en HTML, lo que puede dificultar la modificación o el borrado del manejador de eventos más adelante.

Por estas razones, el uso de atributos "on" se desaconseja en proyectos más grandes y complejos.

Propiedades “on” en HTML

Propiedades “on” en HTML

Las **propiedades "on"** en JavaScript permiten asignar manejadores de eventos directamente a los objetos del DOM mediante código JavaScript, en lugar de hacerlo en el HTML.

Esto separa el código JavaScript del HTML, lo que mejora la organización y la mantenibilidad.

```html
<button id="miBoton">Haz clic aquí</button>
```

```javascript
const boton = document.getElementById("miBoton");
boton.onclick = function() {
  console.log("¡Hola, mundo!");
};
```

Propiedades “on” en HTML -Atencion

Aunque esta técnica mejora la organización en comparación con los atributos "on" en HTML, sigue teniendo algunas **limitaciones**:

Un solo manejador por evento: Puedes asignar solo un manejador de eventos por tipo de evento a un elemento.

Si necesitas asignar múltiples manejadores de eventos al mismo elemento, la notación con propiedades "on" no es la más adecuada.

Propiedades “on” en HTML -Atencion

Dificultad para desvincular eventos: Al igual que con los atributos "on" en HTML, desvincular eventos asignados con propiedades "on" en JavaScript también puede ser complicado.

Problemas de sobreescritura: Si asignas un nuevo manejador de eventos a una propiedad "on" existente, este sobrescribirá al manejador anterior, lo que puede causar conflictos y comportamientos no deseados.

Metodo “addEventListener”

Metodo “addEventListener

El **método "addEventListener"** es la forma más moderna y recomendada de controlar eventos en JavaScript. Con **"addEventListener"**, puedes asignar múltiples manejadores de eventos a un mismo elemento sin afectar los existentes.

Esta flexibilidad lo convierte en la opción preferida para manejar eventos en proyectos de cualquier tamaño.

```
<button id="miBoton">Haz clic aquí</button>
```

```
const boton = document.getElementById("miBoton");
boton.addEventListener("click", function() {
  console.log("¡Hola, mundo!");
});
```

Metodo “addEventListener - ventajas

Flexibilidad: Puedes asignar múltiples manejadores de eventos al mismo elemento sin reemplazar los existentes. Esto permite una mayor modularidad y organización del código.

Mantenimiento y desvinculación: Puedes desvincular eventos en cualquier momento utilizando el método "removeEventListener", lo que facilita el mantenimiento del código y evita conflictos.

Metodo “addEventListener - ventajas

No se sobrescribe: Al agregar nuevos manejadores de eventos con "addEventListener", no se sobrescriben los manejadores existentes. Esto evita problemas de sobreescritura y posibles conflictos.

Buena práctica: Es la forma recomendada por la comunidad de desarrolladores y se adapta mejor a proyectos grandes y complejos.

Método “addEventListener - En acción

Veamos un ejemplo:

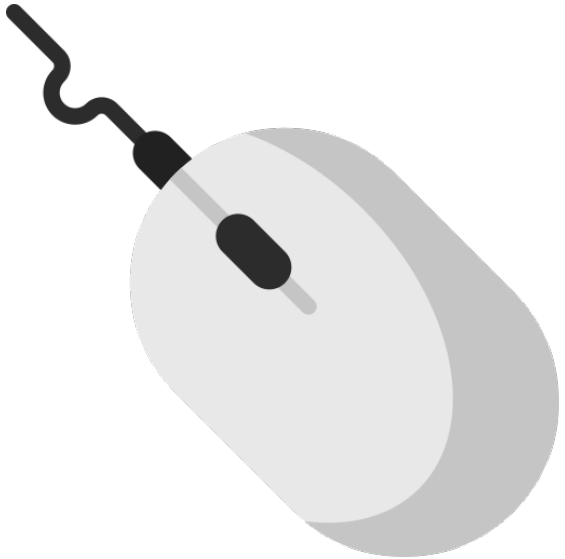
Hemos asignado dos manejadores de eventos diferentes (*"manejarClic" y "manejarSaludo"*) al mismo botón utilizando "addEventListener". Ambas funciones se ejecutarán cuando se haga clic en el botón, lo que demuestra la flexibilidad y utilidad de este método.

```
const boton = document.getElementById("miBoton");

function manejarClic() {
  console.log("¡Clic en el botón!");
}

function manejarSaludo() {
  console.log("¡Hola, mundo!");
}

boton.addEventListener("click", manejarClic);
boton.addEventListener("click", manejarSaludo);
```

Manejo del evento On Click

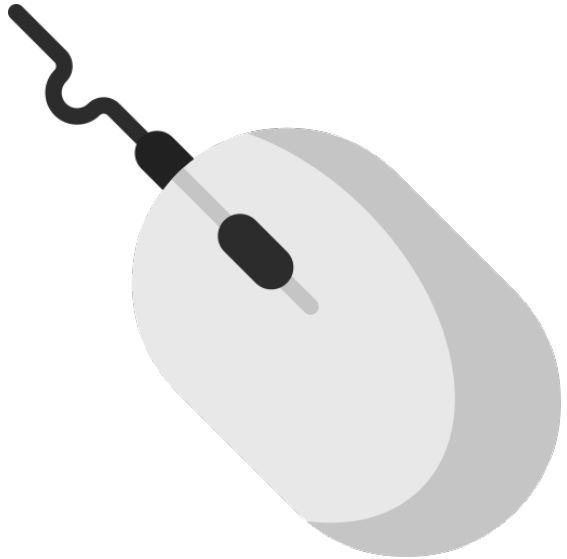
Manejo del evento On Click

El manejo de **eventos de ratón** en JavaScript permite detectar y responder a las interacciones del usuario con el ratón en la página web. Existen varios tipos de eventos de ratón que se pueden utilizar para capturar diferentes acciones realizadas por el usuario.



Manejo del evento On Click

A continuación, describiré los principales eventos de ratón y cómo se pueden utilizar para interactuar con el usuario en una página web.



Evento “click”

El evento **"click"** ocurre cuando el usuario hace clic en un elemento con el botón izquierdo del ratón. Es uno de los eventos más comunes y ampliamente utilizado para activar acciones en respuesta a las interacciones del usuario.

- **Cuándo se activa:** El evento "click" se activa cuando el botón izquierdo del ratón se presiona y luego se suelta sobre el elemento objetivo.
- **Aplicaciones:** Se utiliza comúnmente para botones, enlaces y elementos interactivos que deben responder a la interacción del usuario, como mostrar o ocultar contenido, enviar formularios o navegar a otra página.

```html
<button id="miBoton">Haz clic aquí</button>
```

```javascript
const boton = document.getElementById("miBoton");

boton.addEventListener("click", function() {
  console.log("¡Haz hecho clic en el botón!");
});
```

Evento “dblclick”

El evento **"dblclick"** (doble clic) ocurre cuando el usuario hace doble clic en un elemento con el botón izquierdo del ratón. Es similar al evento "click", pero requiere que el usuario haga dos clics rápidos en lugar de uno solo.

- **Cuándo se activa:** El evento "dblclick" se activa cuando se hace doble clic rápidamente con el botón izquierdo del ratón sobre el elemento objetivo.
- **Aplicaciones:** Se utiliza en situaciones donde se requiere una acción más significativa o especial que el clic único, como ampliar una imagen o realizar una acción importante del usuario.

```html
<button id="miBoton">Haz doble clic aquí</button>
```

```javascript
const boton = document.getElementById("miBoton");

boton.addEventListener("dblclick", function() {
  console.log("¡Haz hecho doble clic en el botón!");
});
```

Evento “mousedown”

El evento **"mousedown"** ocurre cuando el usuario presiona un botón del ratón sobre un elemento. No importa si el botón del ratón aún está presionado o si se ha soltado después.

- **Cuándo se activa:** El evento "mousedown" se activa cuando se presiona cualquier botón del ratón (izquierdo, derecho o central) sobre el elemento objetivo.
- **Aplicaciones:** Se utiliza en casos donde deseas que una acción se active inmediatamente después de que el usuario presione un botón del ratón, independientemente de si lo suelta o no.

```html
<button id="miBoton">Presiona el botón aquí</button>
```

```javascript
const boton = document.getElementById("miBoton");

boton.addEventListener("mousedown", function() {
  console.log("Botón del ratón presionado sobre el botón");
});
```

Evento “mouseup”

El evento **"mouseup"** ocurre cuando el usuario suelta un botón del ratón sobre un elemento.

- **Cuándo se activa:** El evento "mouseup" se activa cuando se suelta cualquier botón del ratón (izquierdo, derecho o central) sobre el elemento objetivo.
- **Aplicaciones:** Se utiliza cuando deseas que una acción ocurra después de que el usuario ha presionado y luego soltado un botón del ratón, como el desbloqueo de una función o activación de una interacción.

```html
<button id="miBoton">Suelta el botón aquí</button>
```

```javascript
const boton = document.getElementById("miBoton");

boton.addEventListener("mouseup", function() {
  console.log("Botón del ratón soltado sobre el botón");
});
```

Evento “mouseover”

El evento **"mouseover"** ocurre cuando el cursor del ratón se coloca sobre un elemento, es decir, cuando el ratón "entra" en el área del elemento objetivo.

- **Cuándo se activa:** El evento "mouseover" se activa cuando el cursor del ratón entra en el área del elemento objetivo.
- **Aplicaciones:** Es comúnmente utilizado para proporcionar retroalimentación visual o mostrar información adicional cuando el usuario coloca el ratón sobre un elemento, como mostrar tooltips, destacar elementos o activar animaciones.

```
<div id="miDiv" style="width: 100px; height:
100px; background-color: lightblue;"></div>
```

```
const miDiv = document.getElementById("miDiv");

miDiv.addEventListener("mouseover", function() {
  console.log("El ratón está sobre el div");
});
```

Evento “mouseout”

El evento **"mouseout"** ocurre cuando el cursor del ratón sale de un elemento, es decir, cuando el ratón "sale" del área del elemento objetivo.

- **Cuándo se activa:** El evento "mouseout" se activa cuando el cursor del ratón sale del área del elemento objetivo.
- **Aplicaciones:** Se utiliza para deshacer o revertir cambios realizados por el evento "mouseover", como ocultar tooltips o restaurar elementos a su estado original.

```
<div id="miDiv" style="width: 100px; height:
100px; background-color: lightblue;"></div>
```

```
const miDiv = document.getElementById("miDiv");

miDiv.addEventListener("mouseout", function() {
  console.log("El ratón ha salido del div");
});
```

Evento “mousemove”

El evento **"mousemove"** ocurre cuando el ratón se mueve sobre un elemento.

- **Cuándo se activa:** El evento "mousemove" se activa continuamente mientras el cursor del ratón se mueve sobre el elemento objetivo.
- **Aplicaciones:** Se utiliza para realizar acciones en tiempo real basadas en el movimiento del ratón, como seguir el cursor, mover elementos o activar animaciones interactivas.

```
<div id="miDiv" style="width: 100px; height:
100px; background-color: lightblue;"></div>
```

```
const miDiv = document.getElementById("miDiv");

miDiv.addEventListener("mousemove",
function(event) {
  console.log("Posición del ratón: ",
event.clientX, event.clientY);
});
```

Ejemplo combinado

En este ejemplo, hemos asignado cada uno de los eventos de ratón mencionados al div "miDiv". Cuando ocurra cada evento, se imprimirá un mensaje correspondiente en la consola, lo que nos permitirá ver los diferentes eventos en acción y cómo se comportan.

```html
<div id="miDiv" style="width: 100px; height:
100px; background-color: lightblue;"></div>
```

```javascript
const miDiv = document.getElementById("miDiv");

miDiv.addEventListener("click", function() {
  console.log("¡Haz hecho clic en el div!");
});

miDiv.addEventListener("dblclick", function() {
  console.log("¡Haz hecho doble clic en el div!");
});

miDiv.addEventListener("mousedown", function() {
  console.log("Presionaste un botón del ratón
sobre el div");
});

miDiv.addEventListener("mouseup", function() {
  console.log("Soltaste un botón del ratón sobre
el div");
});

miDiv.addEventListener("mouseover", function() {
  console.log("El ratón está sobre el div");
});

miDiv.addEventListener("mouseout", function() {
  console.log("El ratón ha salido del div");
});

miDiv.addEventListener("mousemove",
function(event) {
  console.log("Posición del ratón: ",
event.clientX, event.clientY);
});
```

¿Dudas o consultas?

¡Muchas Gracias!