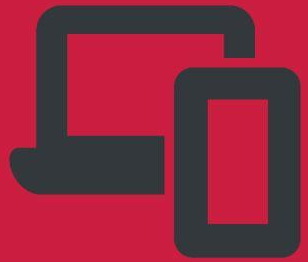
Universidad de SanAndrés



EVENTOS

DigitalHouse > Coding School

/Academy
/Schools
/Corporate Training
/University

EVENTOS

Universidad de SanAndrés

Un **evento** comprende todo aquello que **sucede** en la **ventana** del **navegador** o **está generado** por el **usuario** cuando interactúa con la página web.

Los **eventos** son la herramienta para **incorporar dinamismo** a un sitio web.

Ejemplos de eventos:

- La página terminó de cargar.
- El mouse pasó por arriba de un elemento.
- El usuario hizo click sobre un botón.

Universidad de SanAndrés

TRABAJANDO CON EVENTOS

Cuando trabajamos con eventos tenemos que pensar en 3 elementos clave:

1. ***Dónde*** queremos que ocurra un evento. Para ello debemos definir un **elemento** del DOM sobre el que queremos verificar la ocurrencia de un evento.
2. **Qué evento** debe ocurrir sobre el elemento. Debemos definir qué evento o acción tiene que ocurrir sobre el elemento.
3. **Cuál** será la acción que **sucederá** una vez que el evento ocurra.

Dónde	***Qué***	***Acción***
Etiqueta h1	El usuario haga click	Cambiar el color del texto
Barra de navegación	El usuario pase el mouse por arriba	Cambiar el tamaño de la letra
Una tecla	Presionarla	Mostrar en la consola la tecla presionada

Universidad de SanAndrés

1. IDENTIFICANDO EL ELEMENTO DEL DOM

Usando algunas de las formas vistas para capturar elementos del DOM obtenemos aquel elemento **donde** esperamos detectar que suceda algún evento o acción.

{}

```
<p>Hola mundo</p>

let texto = document.querySelector("p");
```

Universidad de SanAndrés

2. DETECTANDO EL EVENTO

Para detectar un evento utilizaremos una función nativa de javascript que se encarga de "escuchar eventos": ***addEventListener()***

addEventListener() recibe 2 parámetros:

1. El evento al que debe prestar atención
2. Una función anónima (no tiene nombre) a donde escribiremos la acción que queremos tomar cuando el evento suceda. Técnicamente a esta función se llama ***callback***.

{}

```
<p>Hola mundo</a>

let texto = document.querySelector("p");
texto.addEventListenter('evento', function() {
    //Lo que sucede al verificarse el evento
});
```

Universidad de SanAndrés

3. DEFINIENDO LA ACCIÓN POSTERIOR

Dentro de la función anónima (callback) escribiremos el código que queremos ejecutar una vez que el evento suceda.

{}

```
<p>Hola mundo</p>

let texto = document.querySelector("p");
texto.addEventListenter('evento', function() {
    alert('Acaba de pasar algo aquí');
});
```

Universidad de SanAndrés

LISTA DE EVENTOS

Existen múltiples eventos. Algunos de los más conocidos son:

click → *Cuando se hace un click.*
dblclick → *Cuando se hace doble click.*
mouseover → *Cuando se posiciona el mouse sobre un elemento.*
mouseout → *Cuando el mouse sale del elemento,*
mousemove → *Cuando se mueve el mouse.*
scroll → *Cuando se hace scroll sobre la ventana del navegador.*
keydown → *Cuando se presiona una tecla.*
load → *Cuando se carga la ventana del navegador.*
focus → *Cuando se clickea sobre el campo de un formulario.*
blur → *Cuando se sale del campo de un formulario.*
submit → *Cuando se envía el formulario.*

Universidad de SanAndrés

EJEMPLO

En el ejemplo vemos que el código captura un elemento del ***DOM*** usando ***querySelector()*** y el selector de etiqueta ***button.*** *(1)*

Sobre el elemento capturado Implementa 2 eventListeners: uno para detectar un click y otro para detectar cuando el mouse pasa por encima. *(2)*

Dentro de cada función anónima (el callback) se ejecutará una alerta cuando los eventos sucedan. *(3)*

```
let boton = document.querySelector('button');                    (1)

boton.addEventListener('click', function() {                     (2)
     alert('¡Clickeaste el botón!');                             (3)
});

boton.addEventListener('mouseover', function() {                 (2)
     alert('Ahora pasaste el mouse por encima del botón');       (3)
});
```

EVENTOS

Universidad de SanAndrés

Dentro de la función manejadora de eventos ***addEventListener()*** tenemos la posibilidad de obtener información

1. **del propio evento**.
2. del **elemento** sobre el que sucedió.

Universidad de SanAndrés

INFORMACIÓN DEL EVENTO

Para acceder a la **información** del **evento** debemos que pasar un **parámetro** dentro de la **función anónima** (callback) que gestiona la acción.
Javascript entenderá que **ese parámetro representa al evento** sucedido y podremos acceder a su información. Por convención solemos llamarlo ***event*** o simplemente ***e.***

{}

```
window.addEventListener('keyPress', function(e) {
   console.log(e);
   console.log(e.key);
});

//La consola mostrará los datos del evento y por lo tanto podremos saber qué tecla
presionó el usuario accediendo a la propiedad key: e.key
KeyboardEvent {isTrusted: true, key: "d", code: "KeyD", location: 0, ctrlKey: false, ...}
```

Universidad de SanAndrés

INFORMACIÓN DEL ELEMENTO SELECCIONADO

Para poder acceder a la **información** del **elemento** sobre el que se verificó un evento utilizamos la palabra reservada **this** dentro de la **función anónima** (callback) que gestiona la acción.

{}

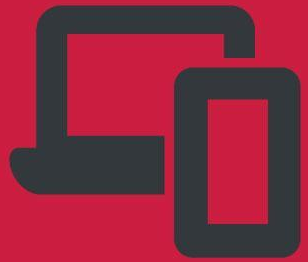
```
let boton = document.querySelector('button');
let mainTitle = document.querySelector('h1’);

boton.addEventListener('click', function() {
  console.log(this);
});
mainTitle.addEventListener('mouseover', function() {
  console.log(this);
});
```

Al hacer **click** sobre el **botón** se verá por consola el elemento <button>.

Al **pasar el mouse** por encima del **título** se verá por consola el elemento <h1>.

Universidad de SanAndrés



EVITANDO ACCIONES POR DEFAULT

DigitalHouse > Coding School

/Academy
/Schools
/Corporate Training
/University

EVITANDO ACCIONES POR DEFAULT

Universidad de SanAndrés

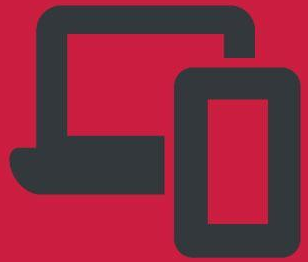
Algunos elemento html tienen asociadas acciones por defecto. Por ejemplo los hipervínculos o el envío de formularios.
La función ***preventDefault()*** **detiene** cualquier comportamiento nativo de un elemento HTML.
preventDefault() se ejecuta sobre el evento que queremos detener; dentro del callback que maneja la acción posterior al evento.

{}

```
let link = document.querySelector('a');

link.addEventListener('click', function(e) {
     e.preventDefault()                 //Evita el comportamiento default del hipervínculo.
     alert('¡Clickeaste el link!');     //Ejecuta la alerta
});
```

Universidad de SanAndrés



¿TODO LISTO?

DigitalHouse > Coding School

/Academy
/Schools
/Corporate Training
/University

TODO LISTO - ONLOAD

Universidad de SanAndrés

onload es un evento que controla que todo el html esté cargado en el navegador.
Si el **HTML no está cargado por completo** las capturas de elementos del DOM pueden fallar. JS intentará obtener un elemento que aún no se cargó y retornará un ***undefined***.

Para asegurarnos de que todo el HTML esté cargado correctamente tenemos 2 estrategias:

1. Colocar la llamada a los archivos js justo antes de cerrar el elemento **<body>**.
2. Implementar en nuestros archivos js un eventListener que controle la carga completa del html para que todo el script se ejecute cuando se haya cargado por completo el objeto ***document*** dentro del objeto ***window.***

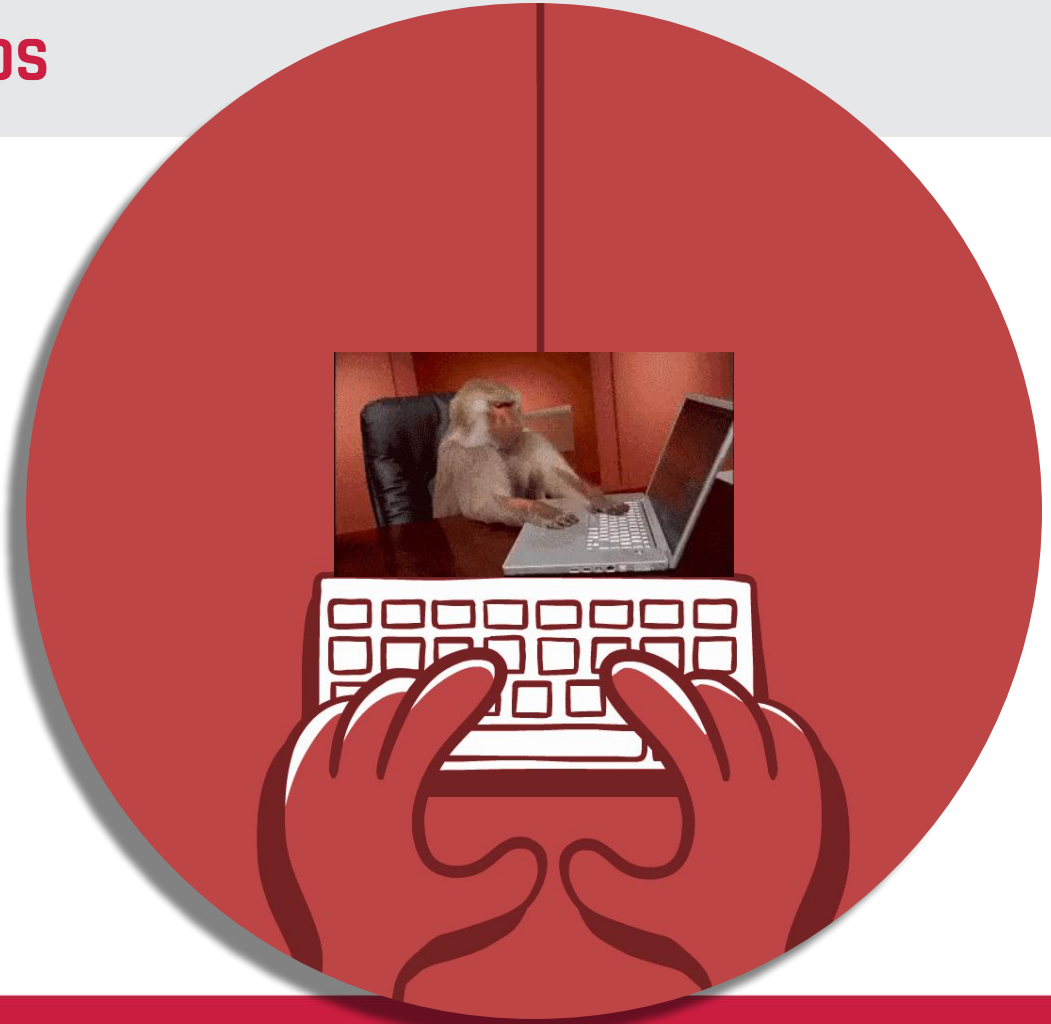
{}

```
window.addEventListener('load', function() {

//Aquí todo el contenido de nuestro archivo javascript.

});
```

PRACTIQUEMOS



DigitalHouse >
Coding School

¡NOS VEMOS LA PRÓXIMA!