Desarrollo Web Frontend

Edición 2021

Universidad Autónoma de Entre Ríos
Facultad de Ciencia y Tecnología – Sede Oro Verde

Eventos en JavaScript

Manejadores de eventos

JSON

Eventos en JavaScript

- Los eventos de JavaScript permiten la interacción entre la aplicación y los usuarios.
- Los eventos pueden ocurrir de diversas maneras. Presionar una tecla o mover el cursor del mouse produce un evento.
- Por otro lado no es obligatorio que el usuario intervenga, ya que por ejemplo el proceso de carga de una página también produce un evento.
- El nombre de los eventos comienzan con el prefijo **on,** seguido del verbo de la acción en inglés. Por ejemplo: **onclick** → al momento de presionar clic con el mouse.

Eventos en JavaScript (Continuación …)

Tipos de eventos.
A continuación se muestran los eventos más importantes definidos por JavaScript.

Evento	Descripción	Elemento para el que está definido
onload	Página cargada completamente	<body>
onmousedown	Pulsar un botón del ratón y no soltarlo	Todos los elementos
onmousemove	Mover el ratón	Todos los elementos
onmouseout	El ratón “sale” del elemento	Todos los elementos
onmouseover	El ratón “entra” en el elemento	Todos los elementos
onmouseup	Soltar el botón del ratón	Todos los elementos
onreset	Inicializar el formulario	<form>
onsubmit	Enviar el formulario	<form>
onselect	Seleccionar un texto	<input>,<select>

Eventos en JavaScript (Continuación …)

Tipos de eventos.

Evento	Descripción	Elemento para el que está definido
onblur	El elemento pierde el foco	<button>,<input>,<label>, <select>,<textarea>, <body>
onchange	El elemento ha sido modificado	<input>,<select>, <textarea>
onclick	Pulsar una vez con el ratón	Todos los elementos
ondblclick	Pulsar dos veces con el ratón	Todos los elementos
onfocus	El elemento obtiene foco	<button>,<input>,<label>, <select>,<textarea>, <body>
onkeydown	Pulsar una tecla y no soltarla	Elementos de formulario y <body>
onkeypress	Pulsar una tecla	Elementos de formulario y <body>
onkeyup	Soltar una tecla pulsada	Elementos de formulario y <body>

~~Eventos en JavaScript~~

Manejadores de eventos

JSON

Manejadores de eventos

Los eventos definidos en JavaScript pueden ser gestionados de diferentes maneras, a continuación vemos cada uno de ellos:

- Mediante un atributo HTML precedido de **on** que llama a la función. (Menos recomendado)

- Mediante propiedades precedidas de **on** que contienen una función.

- A través de la escucha de eventos utilizando listeners.

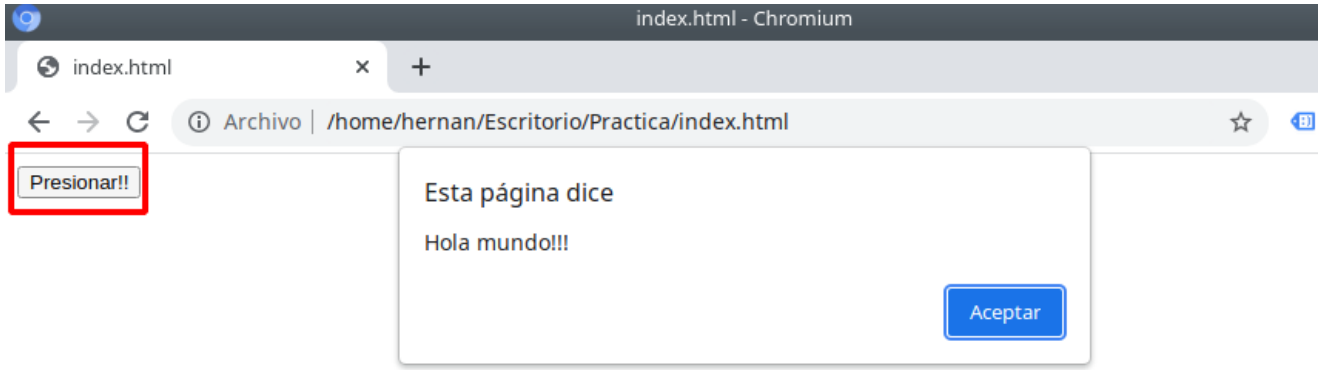
Manejadores de eventos (Continuación …)

Manejador de eventos mediante HTML.

Es la forma más sencilla (y menos profesional) de crear eventos a partir de atributos en los elementos HTML. El código se encuentra definido dentro del valor del atributo del elemento HTML.

Veamos un ejemplo:

```
<button onclick="alert('Hola mundo!!!');" >Presionar!!</button>
```



Manejadores de eventos (Continuación …)

Manejador de eventos mediante HTML.

También es posible enlazar un evento a funciones de JavaScript.

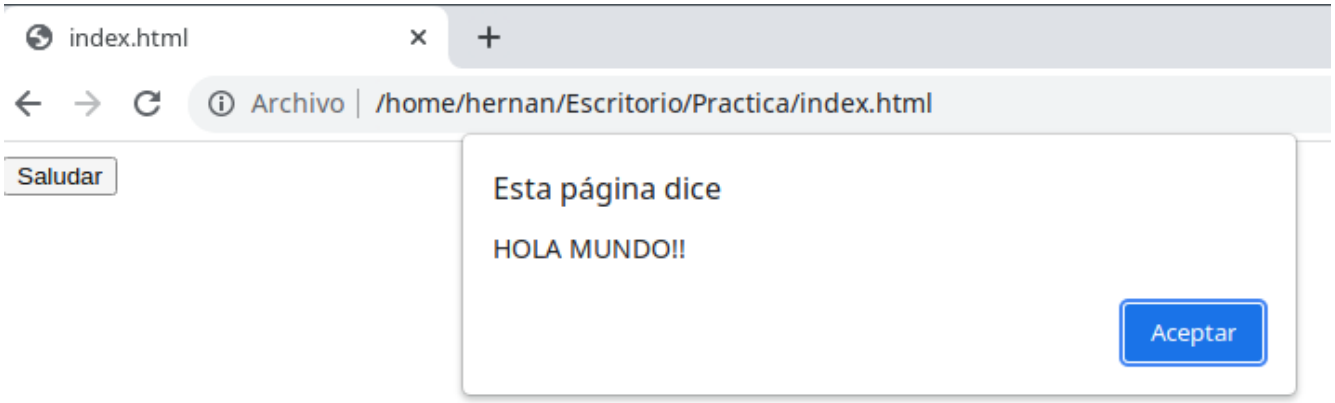
Por ejemplo:

index.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <script src="js/index.js"></script>
</head>
<body>

  <button onclick="saludar();" >Saludar</button>

</body>
</html>
```

js > index.js

```js


function saludar(){
    alert("HOLA MUNDO!!");
}

```



Manejadores de eventos (Continuación …)

Manejador de eventos mediante HTML.

Si bien resulta sencillo definir un evento como en el ejemplo anterior, utilizar este tipo de declaración tiene algunas desventajas:

- Se mezcla código HTML con código JavaScript. (Idealmente deberían estar separados)
- Si la sintaxis del evento **onClick** es muy extensa, dificulta mucho la legibilidad y el mantenimiento del mismo.
- No es posible reutilizar las acciones del evento **onClick** en otros elementos.
- Si se utilizan funciones se soluciona el punto anterios, pero los elementos que la utilicen dependerían del nombre de dicha función.

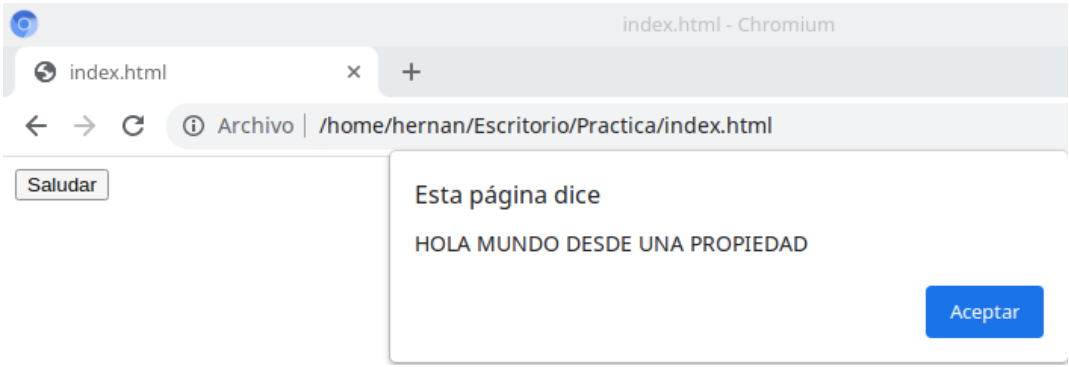
Manejadores de eventos (Continuación …)

Manejador de eventos mediante propiedades.

En esta caso el comportamiento de un evento se define desde JavaScript.
Para esto se utiliza alguna de las propiedades precedidas por **on** del elemento HTML, a la cual le asignaremos el código necesario.

Por ejemplo:

```
<body>

  <button id="btn">Saludar</button>

  <script>
      const button = document.querySelector('#btn');
      button.onclick = function () {
          alert("HOLA MUNDO DESDE UNA PROPIEDAD");
      }
  </script>
</body>
</html>
```



El nombre del evento siempre va en **minúsculas** ya que se trata de un atributo

Manejadores de eventos (Continuación …)

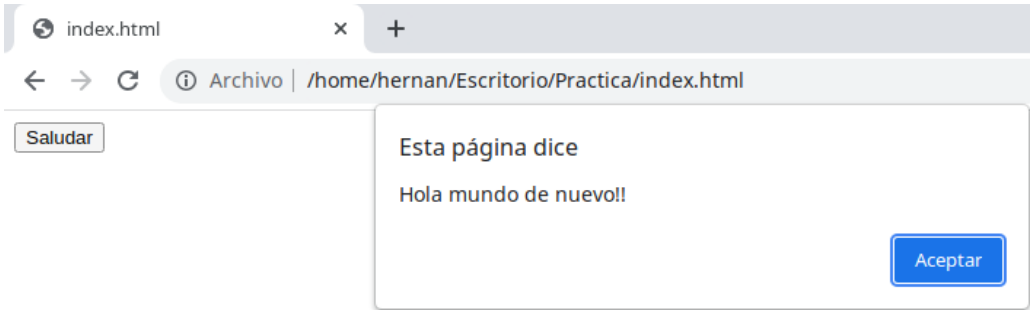
Manejador de eventos mediante listeners.

La manera más versátil de gestionar eventos en JavaScript es utilizando el método **.addEventListener(<string> evento, <function> func)**.

Este método pone en escucha al elemento HTML hasta que se lance el evento definido como primer parámetro y ejecute la función indicada en el segundo parámetro.

Veamos un ejemplo:

```
<body>

  <button id="btn">Saludar</button>

  <script>
      const button = document.querySelector('#btn');
      button.addEventListener('click', function() {
        alert("HOLA MUNDO DE LISTENER");
      });

      //Otra manera
      const saludar = () => alert('Hola mundo de nuevo!!');
      button.addEventListener('click', saludar);

  </script>
</body>
```



Para este caso se omite el prefijo **on** del evento

Manejadores de eventos (Continuación …)

Manejador de eventos mediante listeners.

Además existe la posibilidad de eliminar un listener previamente añadido. En este caso debemos hacer uso del método **.removeEventListener()** y teniendo la función externa en una constante.

```
<body>

  <button id="btn">Saludar</button>

  <script>
      const button = document.querySelector('#btn');
      const saludo = function() {
        alert("Hola desde funcion en constante");
      }

      button.addEventListener('click', saludo);     //Se crea el listener
      button.removeEventListener('click', saludo);  //Se elimina el listener

  </script>
</body>
```

Manejadores de eventos (Continuación …)

El objeto event
En el momento que se dispara un evento no solo se ejecuta la función asociada, sino que también existe información relativa al evento producido.

Entonces a través del objeto **event** podemos saber por ejemplo que tecla fue presionada, en qué posición se ubica el mouse al momento del clic o incluso que elemento a producido el evento.

- El objeto **event** es el encargado de proporcionar al navegador toda la información asociada al evento.

- El estándar DOM define que el objeto **event** debe ser el único parámetro que se debe pasar a una función encargada de procesar un evento.

Manejadores de eventos (Continuación …)

Accediendo al objeto event

Pasando como parámetro de una función el objeto **event** y poder recibirlo y acceder a sus atributos.

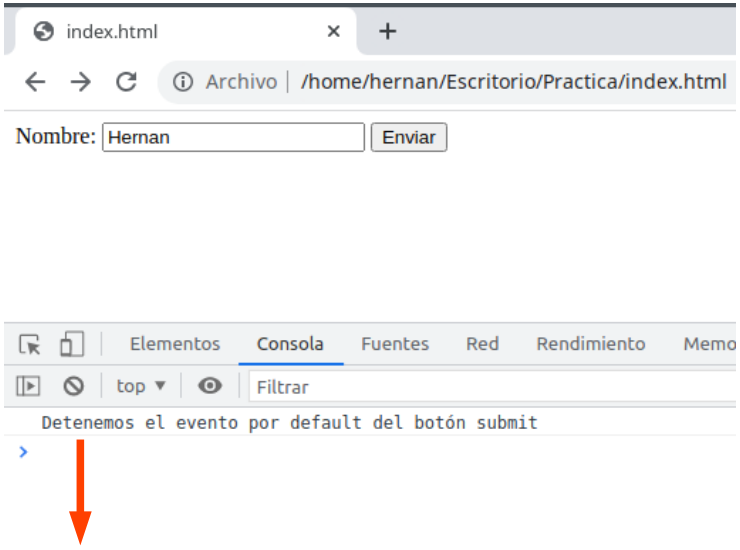
```
<body>

  <button id="btn">Saludar</button>

  <script>
      const button = document.querySelector('#btn');

      //acceder al objeto event
      button.onclick = function (event){
        console.log(event);
      }

      //otra forma de acceder al objeto event
      button.addEventListener('click', function(event) {
        console.log(event);
      })

      //otra forma más
      const mostrarEvent = function(event){
        console.log(event);
      }
      button.addEventListener('click', mostrarEvent);

  </script>
</body>
```

Manejadores de eventos (Continuación …)

El evento **.preventDefault()** es un método asociado al objeto **event,** donde si existe una acción predeterminada que pertenece al evento entonces la misma no ocurrirá.

Por ejemplo al presionar un botón de tipo **submit** en un formulario el evento **preventDefault()** evita la acción por defecto de enviar el mismo. Veamos un ejemplo:

```
<body>

  <form action="index.php" method="get">
    <label>Nombre: </label>
    <input type="text" name="nombre" placeholder="Ingrese su nombre" value="">
    <button id="btn" type="submit">Enviar</button>
  </form>

  <script>
      const button = document.querySelector('#btn');

      button.onclick = function (event){
        event.preventDefault();
        console.log('Detenemos el evento por default del botón submit');
      }
  </script>
</body>
```



Se detiene el envío del formulario a **index.php**

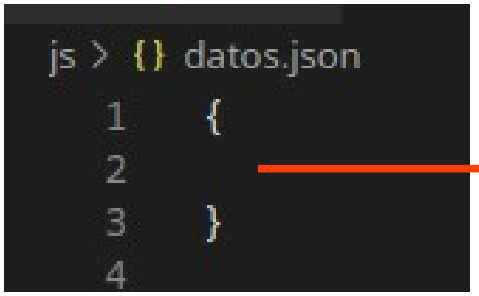
~~Eventos en JavaScript~~

~~Manejadores de eventos~~

JSON

JSON

- Sus siglas significan **JavaScript Object Notation**.
- Es una estructura o notación específica de datos.
- Compatible de forma nativa con JavaScript.
- Se basa en la sintaxis de JavaScript para crear objetos.
- Soporta los tipos de dato **string, number, object, array, boolean** o **null**.



En este caso se trata de un objeto vacío

Las llaves **{ }** representan la sintaxis mínima de un archivo JSON.

JSON (Continuación …)

Utilizando JSON en JavaScript

La sintaxis de JSON es similar a la definición de objetos en JavaScript.

Definición de un objeto en JS

```
js > JS index.js > ...
1
2   let heroe = {
3       tipo: "Guerrero",
4       energia: 250,
5       fuerza: 400,
6       vida: 500,
7       habilidades: {
8           saltar : true,
9           correr : true,
10          volar  : false
11      }
12  }
```

Definición de un objeto JSON

```
js > {} datos.json > ...
1   {
2       "tipo": "Guerrero",
3       "energia": 250,
4       "fuerza": 400,
5       "vida": 500,
6       "habilidades": {
7           "saltar" : true,
8           "correr" : true,
9           "volar"  : false
10      }
11  }
```

JSON (Continuación …)

Utilizando JSON en JavaScript

Existen algunos métodos en JavaScript que permiten convertir un objeto JavaScript en un JSON y viceversa.

- **JSON.parse(str)** → convierte el texto **str** (debe ser un JSON válido) a un objeto y lo devuelve.

- **JSON.stringify(obj)** → convierte un objeto Javascript **obj** a su representación JSON y lo devuelve.

JSON (Continuación …)

Convirtiendo un objeto JavaScript a JSON y viceversa

Ejemplo:

```
js > JS index.js > ...
1
2   let heroe = {
3       tipo: "Guerrero",
4       energia: 250,
5       fuerza: 400,
6       vida: 500,
7       habilidades: {
8           saltar : true,
9           correr : true,
10          volar  : false
11      }
12  }
13
14
15  const jsonHeroe = JSON.stringify(heroe);
16  console.log(jsonHeroe);
```

Object → JSON

Salida del filtro

```
{"tipo":"Guerrero","energia":250,"fuerza":400,"vida":500,"habilidades":{"saltar":true,"correr":true,"volar":false}}
```

String en formato JSON

JSON (Continuación …)

Convirtiendo un objeto JavaScript a JSON y viceversa

Ejemplo:

JSON → Object

```
const jsonHeroe = '{"tipo":"Guerrero","energia":250,"fuerza":400,"vida":500,"habilidades":{"saltar":true,"correr":true,"volar":false}}'
let heroe = JSON.parse(jsonHeroe);

console.log(heroe);
```

¿Consultas?