



# **Uso de JavaScript (II): Eventos y formularios**

**Departamento de Lenguajes y Sistemas  
Informáticos**  
**Universidad de Sevilla**

*Daniel Ayala, Carlos Arévalo, José Calderón, Margarita Cruz, Inma Hernández, David Ruiz*

**UNIVERSIDAD DE SEVILLA**

# Contenidos

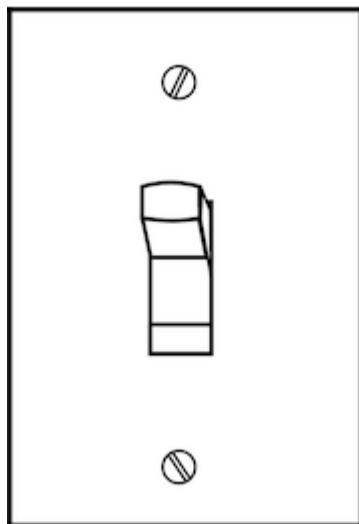
---

1. Introducción
2. Manejo de eventos en JavaScript
3. Ejemplos
4. El objeto Event
5. Validación de formularios

# Eventos de programación

---

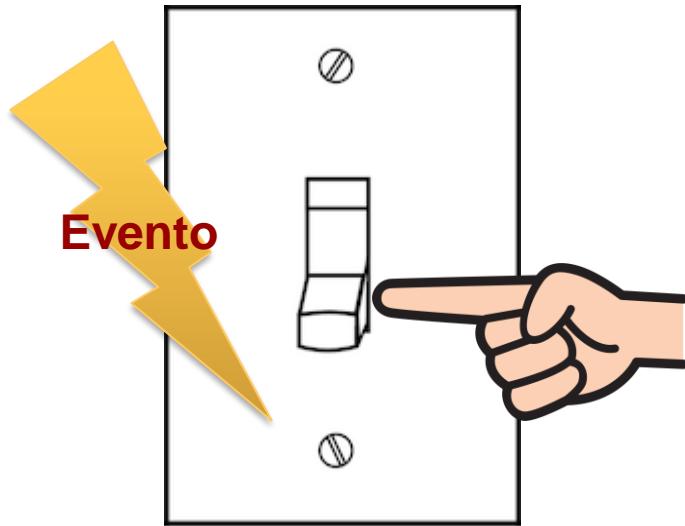
- ◆ Son “cosas” que les pasan a los elementos del lenguaje



# Eventos de programación

---

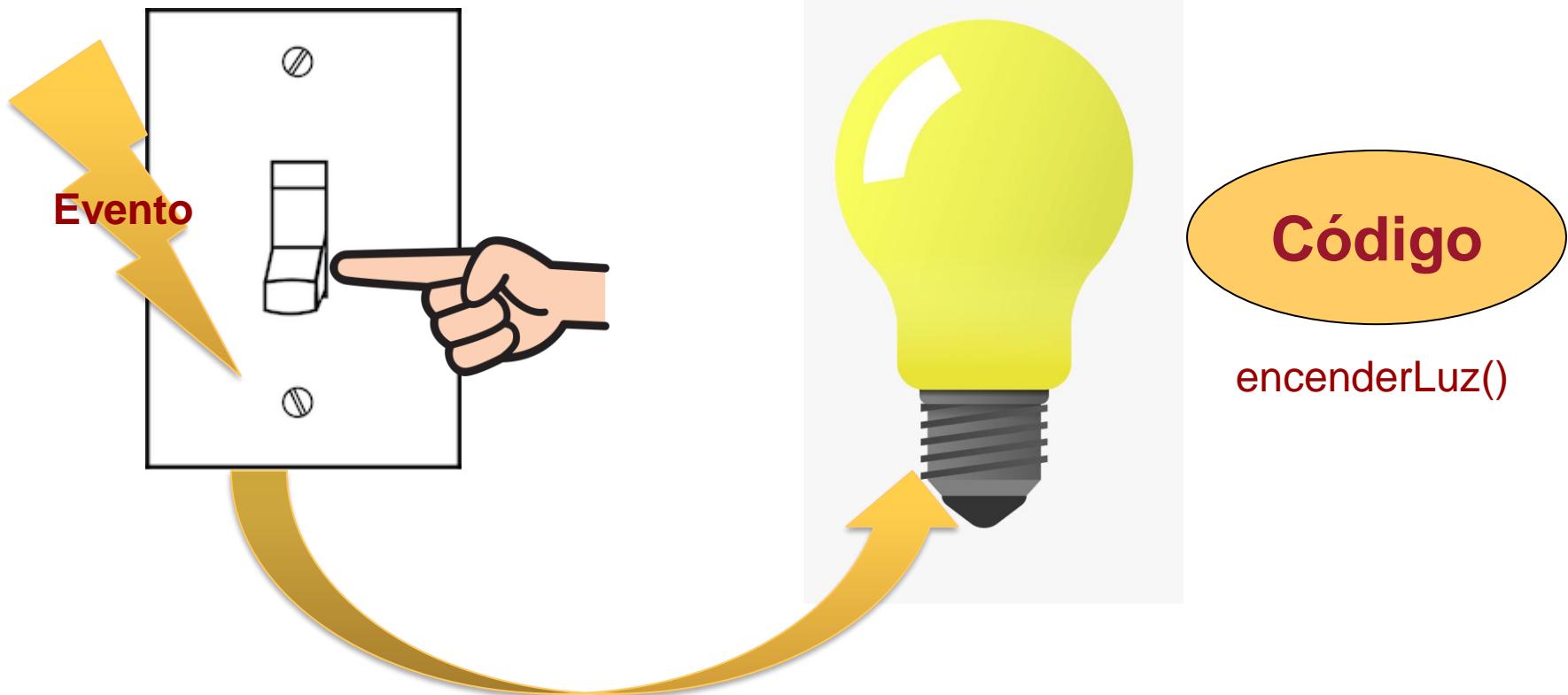
- ◆ Son “cosas” que les pasan a los elementos del lenguaje
- ◆ El lenguaje permite reaccionar a esas “cosas” ejecutando código



# Eventos de programación

---

- ◆ Son “cosas” que les pasan a los elementos del lenguaje
- ◆ El lenguaje permite reaccionar a esas “cosas” ejecutando código



# Eventos en HTML

---

- ◆ En el caso de las páginas web, los eventos son “cosas” que les ocurren a los elementos HTML. Puede ser
  - ◆ A un único elemento
  - ◆ A un conjunto de elementos
  - ◆ A la página entera
- ◆ Los eventos se producen en el contexto del navegador que está mostrando la página, por lo que también afectan al propio navegador

# Tipos de eventos

---

- ◆ Existe una gran variedad de tipos de eventos, según el elemento de que se trate. Algunos ejemplos comunes de cosas que se pueden controlar mediante eventos:
  - ◆ Qué hacer cada vez que se carga la página
  - ◆ Qué hacer cada vez que se cierra la página
  - ◆ Qué hacer cuando el usuario pulsa un elemento (o pasa el ratón por encima de él)
  - ◆ Qué hacer cuando se envía un formulario (por ejemplo, validar los datos que ha introducido el usuario antes de enviar a backend)
  - ◆ Qué hacer cuando el usuario introduce un valor usando el teclado (por ejemplo, para hacer la validación a medida que el usuario escribe, en vez de esperar a enviar el formulario)
  - ◆ Qué hacer cuando el usuario redimensiona la ventana del navegador
- ◆ Lista completa de tipos de eventos:
  - ◆ [https://www.w3schools.com/jsref/dom\\_obj\\_event.asp](https://www.w3schools.com/jsref/dom_obj_event.asp)
  - ◆ <https://developer.mozilla.org/en-US/docs/Web/Events>

# Categorías de eventos

---

- ◆ Según el tipo de elemento que emite el evento, distinguimos:
  - ◆ Eventos del objeto **window**
    - ◆ redimensionar la ventana
  - ◆ Eventos del objeto **window.screen**
    - ◆ cambiar la orientación del dispositivo
  - ◆ Eventos del objeto **document**
    - ◆ carga de la página, interacción con el usuario, salida de la página
  - ◆ Objetos del árbol **DOM**
    - ◆ Interacciones o modificaciones de los elementos
  - ◆ Objetos **XMLHttpRequest**
    - ◆ Para peticiones a través de la red (lo veremos más adelante)
  - ◆ Objetos **multimedia** (audio, video)
    - ◆ Cambio en el estado de la reproducción

# Contenidos

---

1. Introducción
- 2. Manejo de eventos en JavaScript**
3. Ejemplos
4. El objeto Event
5. Validación de formularios

# Como manejar los eventos

---

- ◆ Añadir un manejador al elemento
- ◆ Manipular el DOM para añadir el manejador al elemento
- ◆ Añadir un listener

# Event handlers – atributo manejador

---

- ◆ Para definir cómo se debe reaccionar a un evento que afecta a un determinado elemento, se añade a dicho elemento un manejador de eventos (**event handler**), con la siguiente sintaxis:

```
<elemento manejador='código JavaScript'>
```

- ◆ El código JavaScript para manejar un determinado evento puede ser muy largo como para añadirlo directamente al valor del atributo. Lo habitual es encapsularlo en una función, y llamarla desde el manejador:

```
<elemento manejador='function()'>
```

- ◆ Asociar un trozo de código (o función) a un evento se conoce como registrar un manejador de eventos (*registering an event handler*).

# Event handlers – manipulación del DOM

---

- ◆ Otra forma de definir los handlers es obteniendo una referencia al elemento como objeto. Cada objeto tiene una propiedad por cada tipo de evento:

```
let btn = document.getElementById("boton");
btn.onclick = function() {
    //código de la función
}
```

- ◆ O, si se ha definido la función previamente, asignando directamente a la propiedad el nombre de la función

```
function manejador() {
    //código de la función
}
btn.onclick = manejador;
```

En este caso, se usa el nombre del tipo de evento con el prefijo 'on' (onclick, onmouseover, ...)

# Event Listeners

---

- ◆ También se puede definir el comportamiento cuando se produce un evento con un *event listener*:

```
element.addEventListener(evento, función);
```

- ◆ Ejemplo:

```
btn.addEventListener("click", function(){
    alert("Event listener");
});
```

En este caso, se usa el nombre del tipo de evento sin 'on' (click, mouseover, ...)

# Event handlers vs Event listeners

---

- ◆ A los manejadores de eventos a veces se le llama escuchadores de eventos (*event listeners*), dado que en ambos casos se trata de **código que se ejecuta cuando se produce un evento**
- ◆ Conceptualmente no son lo mismo
  - ◆ El listener está pendiente de que ocurra el evento (el listener *notifica* al handler)
  - ◆ Un handler es la combinación entre
    - ◆ El listener que informa cuando se produce el evento
    - ◆ El código que se ejecuta como respuesta al evento (*callback function*)
- ◆ En la práctica, se usan ambos términos de forma intercambiable, aplicando una **metonimia** (se designa al listener por el nombre del handler para el que “trabaja”)

# Diferencias entre handlers y listeners

---

- ◆ Si se añaden varios handlers al mismo evento, **sólo se ejecutará el último** (se sobreescreiben)
- ◆ Si se añaden varios listeners al mismo evento, **todos ellos se ejecutarán** cuando se produzca el evento
- ◆ Lo mismo si se añade un handler y varios listeners
- ◆ Al método addEventListener() se le puede añadir un tercer parámetro booleano opcional que indica si el código se debe ejecutar en la fase de captura o de bubbling.
- ◆ Al asociar una función a un evento mediante el método addEventListener(), la función recibirá automáticamente un parámetro “event” que representa al evento y contiene propiedades útiles.

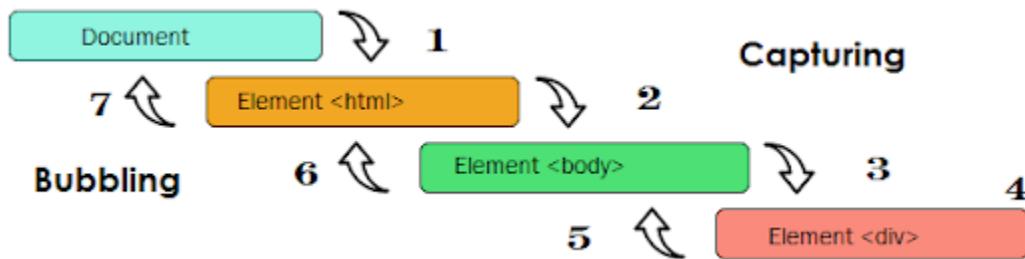
# Modos de propagación de eventos

---

- ◆ En HTML, hay una estructura anidada de elementos
  - ◆ Ej: Un <p> dentro de un <div>, dentro de <body>...
- ◆ Si hay varios elementos anidados, y cada uno de ellos tiene un manejador distinto para un tipo de evento (ej: click), ¿En qué orden se ejecuta el código asociado a los manejadores?
- ◆ Hay dos modos de propagación
  - ◆ Capturing: El orden es del elemento más externo al más interno
    - ◆ En el ejemplo, primero se ejecuta el código del manejador de <body>, luego el de <div> y luego el de <p>
  - ◆ Bubbling: El orden es del elemento más interno al más externo
    - ◆ En el ejemplo, primero se ejecuta el código del manejador de <p>, luego el de <div> y luego el de <body>

# Modos de propagación de un evento

---



# Contenidos

---

1. Introducción
2. Manejo de eventos en JavaScript
- 3. Ejemplos**
4. El objeto Event
5. Validación de formularios

# Eventos en JavaScript - Ejemplo

The screenshot shows a browser window titled "Ejemplos con eventos". The main content area displays the heading "Botón sin evento" and a button labeled "Saluda". The browser's developer tools are open, specifically the Elements and Console tabs.

**Elements Tab:** Shows the DOM structure:

```
<html>
  <head>...</head>
  ...<body> == $0
    <h1>Botón sin evento</h1>
    <button>Saluda</button>
  </body>
</html>
```

**Console Tab:** Shows the following output:

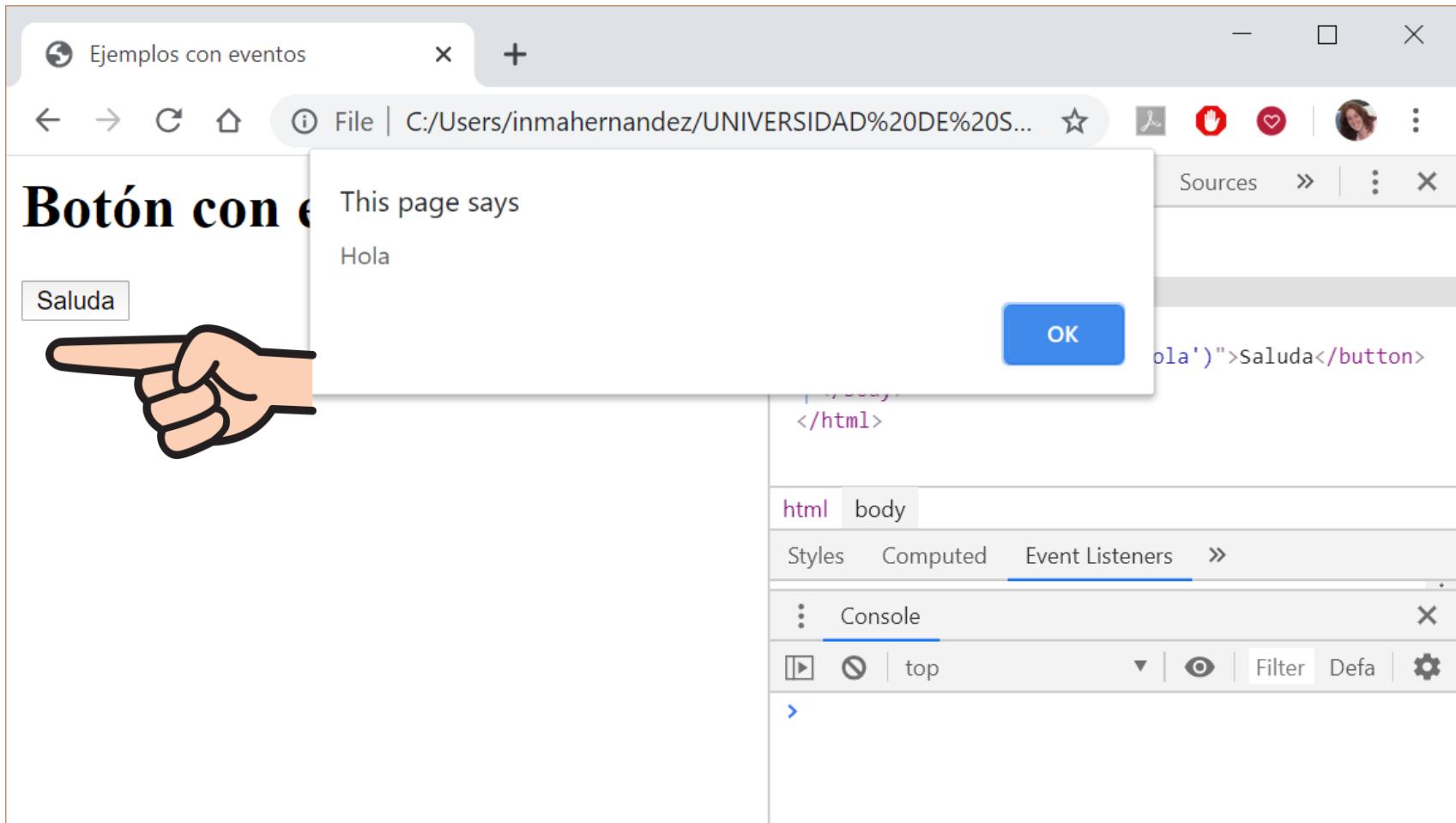
```
>
```

# Eventos en JavaScript - Ejemplo

The screenshot shows a browser window titled "Ejemplos con eventos". The main content area displays the heading "Botón con evento" and a button labeled "Saluda". The browser's developer tools are open, specifically the "Elements" tab. A red oval highlights the line of code for the button's onclick event handler: "button onclick="alert('Hola')">Saluda</button>". The "Event Listeners" tab in the developer tools is also visible.

```
<html>
  <head>...</head>
  ...<body> == $0
    <h1>Botón con evento</h1>
    <button onclick="alert('Hola')">Saluda</button>
  </body>
</html>
```

# Eventos en JavaScript - Ejemplo



# Eventos en JavaScript - Ejemplo

The screenshot shows a browser window titled "Ejemplos con eventos". The main content area displays an HTML page with the title "Botón con evento" and a single button labeled "Saluda". The browser's developer tools are open, specifically the "Sources" tab, which is displaying the source code for "eventos.html". The code includes an "html" tag with a "head" section containing a title and a script tag, and a "body" section containing an "h1" and a "button" element with an "onclick" attribute pointing to the "saludar" function. Below the "Sources" tab, there is a large red box containing the actual JavaScript code for the "saludar" function:

```
'use strict';

function saludar() {
    let fechaHoy = new Date();
    alert('Hola, son las: ' + fechaHoy.toLocaleString());
}
```

The developer tools also show the "Console" tab at the top, and a "Scope" tab in the bottom right corner of the red box.

# Eventos en JavaScript - Ejemplo

The screenshot shows a browser window titled "Ejemplos con eventos". The main content area displays the text "Botón con evento" and a button labeled "Saluda". A hand icon is pointing at the "Saluda" button. A tooltip-like overlay says "This page says" followed by "Hola, son las: 3/9/2020, 7:54:24 PM". An "OK" button is visible on the right side of the overlay. In the bottom right corner of the browser window, there is a developer tools panel showing the code for "eventos.html" and a console output.

This page says

Hola, son las: 3/9/2020, 7:54:24 PM

OK

```
eventos.html
6   </head>
7
8   <body>
9     <h1>Botón con evento</h1>
10    <button onclick="saludar()" id="boton">Saluda</button>
11
12
13 </html>
```

{ Line 4, Column 21

Scope Watch

Console

at file:///C:/Users/inmahernandez/UNIVERSIDAD%20DE%20SEVILLA/ISSI2%20(19-20)%20-%20Teor%C3%ADa/ejemplos%20html/eventos.js:3:33

# Varios handlers

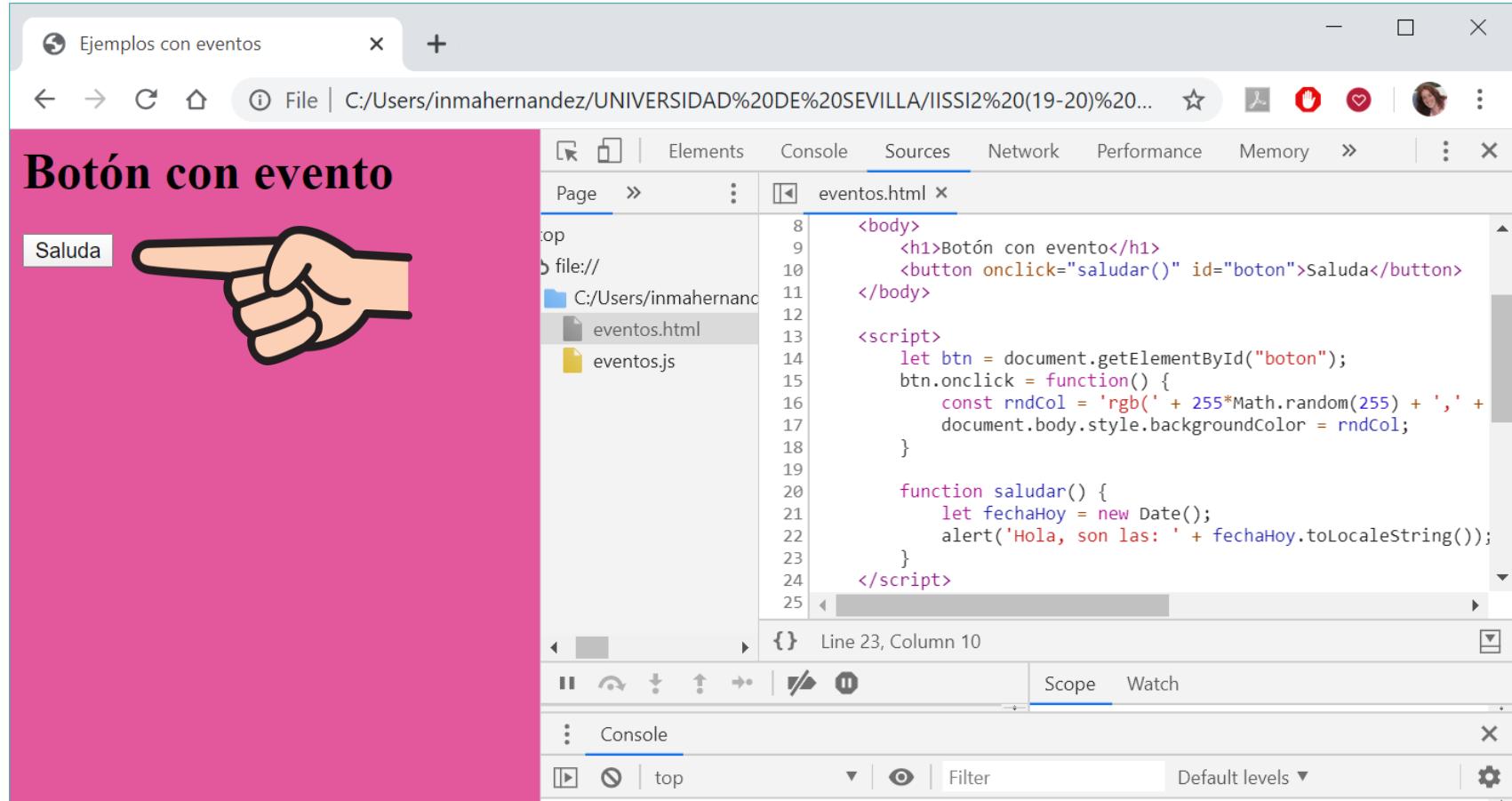
The screenshot shows a browser window titled "Ejemplos con eventos". On the left, there is a button labeled "Saluda". In the center, there is a heading "Botón con evento" and the button "Saluda". On the right, the browser's developer tools are open, specifically the Sources tab. It displays the code for "eventos.html" and "eventos.js". A yellow callout box with the text "Función anónima" points to the line of code in "eventos.js" where an anonymous function is assigned to the "onclick" event of the button.

```
let btn = document.getElementById("boton");
btn.onclick = function() {
  const rndCol = 'rgb(' + 255*Math.random(255) + ',' + 2
55*Math.random(255) + ',' + 255*Math.random(255) + ')';
  document.body.style.backgroundColor = rndCol;
}

function saludar() {
  let fechaHoy = new Date();
  alert('Hola, son las: ' + fechaHoy.toLocaleString());
}
```

Función anónima

# Varios handlers



The screenshot shows a browser window titled "Ejemplos con eventos". The main content area displays a pink background with a hand icon pointing to a button labeled "Saluda". The browser's developer tools are open, specifically the "Sources" tab, which is displaying the code for "eventos.html". The code includes an

# Más clicks...

The image shows four overlapping browser windows, each displaying a page titled "Botón con evento". The windows are colored green, purple, blue, and dark green from top to bottom. Each window has a "Saluda" button. The background color of the windows changes randomly when the button is clicked. The bottom-most window's source code is visible in the developer tools' Sources tab:

```
8 <body>
9   <h1>Botón con evento</h1>
10  <button onclick="saludar()" id="boton">Saluda</button>
11 
12 
13 <script>
14   let btn = document.getElementById("boton");
15   btn.onclick = function() {
16     const rndcol = 'rgb(' + 255*Math.random(255) + ',' +
17       document.body.style.backgroundColor = rndCol;
18   }
19 
20   function saludar() {
21     let fechaHoy = new Date();
22     alert('Hola, son las: ' + fechaHoy.toLocaleString());
23   }
24 </script>
25 
```

The developer tools also show the "Console" tab at the bottom.

**¿Y la función saluda?**

# Event listeners

The screenshot shows a browser window titled "Ejemplos con eventos". On the left, there is a button labeled "Saluda". In the developer tools' Sources tab, the file "eventos.html" is open, displaying the following code:

```
let btn = document.getElementById("boton");
btn.addEventListener('click', cambiarColor, false);
function cambiarColor() {
  const rndCol = 'rgb(' + 255*Math.random(255) + ',' + 255*Math.random(255) + ',' + 255*Math.random(255) + ')';
  document.body.style.backgroundColor = rndCol;
}

function saludar() {
  let fechaHoy = new Date();
  alert('Hola, son las: ' + fechaHoy.toLocaleString());
}
```

The developer tools also show the DOM tree on the left, with "eventos.html" selected. The "Sources" tab is active, and the "Console" tab is visible at the top.

# Event listeners

The screenshot shows a browser window titled "Ejemplos con eventos". The main content area displays the text "Botón con evento" and "This page says Hola, son las: 3/9/2020, 8:18:36 PM". A hand icon is pointing at a button labeled "Saluda". An "OK" button is visible in a modal. The developer tools sidebar shows the file "eventos.js" with the following code:

```
13 <script>
14   let btn = document.getElementById("boton");
15   btn.addEventListener('click', cambiarColor, false);
16   function cambiarColor() {
17     const rndCol = 'rgb(' + 255*Math.random(255) + ', ' +
18       document.body.style.backgroundColor = rndCol;
19   }
20
21   function saludar() {
22     let fechaHoy = new Date();
23     alert('Hola, son las: ' + fechaHoy.toLocaleString());
24   }
25
```

The status bar indicates "Line 23, Column 10". The bottom of the screen shows the browser's toolbar and a "Console" tab in the developer tools.

# Varios Event Listeners

The screenshot shows a browser window titled "Ejemplos con eventos". The main content area displays a heading "Botón con evento" and a button labeled "Saluda". The browser's developer tools are open, specifically the "Sources" tab, which is displaying the file "eventos.html". The code in the file is as follows:

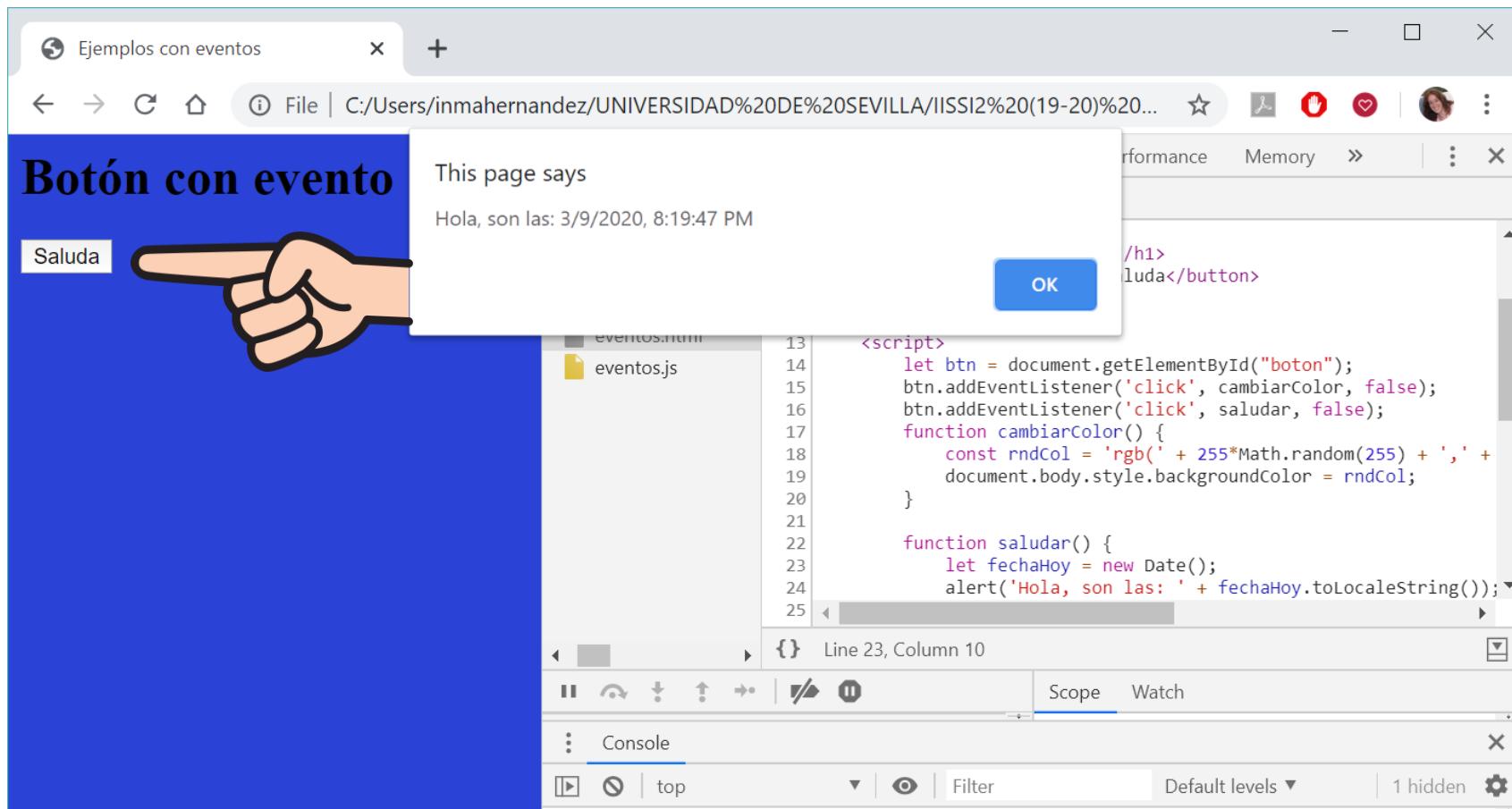
```
let btn = document.getElementById("boton");
btn.addEventListener('click', cambiarColor, false);
btn.addEventListener('click', saludar, false);
function cambiarColor() {
  const rndCol = 'rgb(' + 255*Math.random(255) + ', ' +
  255*Math.random(255) + ',' + 255*Math.random(255) + ')';
  document.body.style.backgroundColor = rndCol;
}

function saludar() {
  let fechaHoy = new Date();
  alert('Hola, son las: ' + fechaHoy.toLocaleString());
}
```

The "Sources" tab also shows the file "eventos.js" which contains the following code:

```
let btn = document.getElementById("boton");
function cambiarColor() {
  let rndCol = 'rgb(' + 255*Math.random(255) + ', ' +
  255*Math.random(255) + ',' + 255*Math.random(255) + ')';
  document.body.style.backgroundColor = rndCol;
}
function saludar() {
  let fechaHoy = new Date();
  alert('Hola, son las: ' + fechaHoy.toLocaleString());
}
```

# Varios Event Listeners



The screenshot shows a browser window titled "Ejemplos con eventos". The main content area displays the text "Botón con evento" and "This page says Hola, son las: 3/9/2020, 8:19:47 PM". A hand icon is pointing at a blue button labeled "Saluda". An "OK" button is visible in a modal. The developer tools are open, showing the file "eventos.js" with the following code:

```
13 <script>
14     let btn = document.getElementById("boton");
15     btn.addEventListener('click', cambiarColor, false);
16     btn.addEventListener('click', saludar, false);
17     function cambiarColor() {
18         const rndCol = 'rgb(' + 255*Math.random(255) + ',' +
19                     document.body.style.backgroundColor = rndCol;
20     }
21
22     function saludar() {
23         let fechaHoy = new Date();
24         alert('Hola, son las: ' + fechaHoy.toLocaleString());
25     }

```

The code uses event listeners to change the background color of the page and display a greeting message when the button is clicked.

# Event Listeners para distintos tipos de evento

The screenshot shows a browser window titled "Ejemplos con eventos". The main content area displays a heading "Botón con evento" and a button labeled "Saluda". The browser's developer tools are open, specifically the Sources tab, which is currently selected. The code editor shows the HTML file "eventos.html" and the JavaScript file "eventos.js". The "eventos.html" file contains a simple button element with an onclick attribute pointing to the "saludar" function. The "eventos.js" file contains the following code:

```
let btn = document.getElementById("boton");
btn.addEventListener('click', cambiarColor, false);
btn.addEventListener('mouseout', saludar, false);

function cambiarColor() {
    const rndCol = 'rgb(' + 255*Math.random(255) + ', ' +
    255*Math.random(255) + ',' + 255*Math.random(255) + ')';
    document.body.style.backgroundColor = rndCol;
}

function saludar() {
    let fechaHoy = new Date();
    alert('Hola, son las: ' + fechaHoy.toLocaleString());
}
```

The developer tools also show the DOM tree on the left, with "eventos.html" selected. The Network, Performance, and Memory tabs are visible at the top of the tools panel.

# Event Listeners para distintos tipos de evento

The screenshot shows a browser window titled "Ejemplos con eventos". The main content area displays a dark purple background with a hand icon pointing at a button labeled "Saluda". The browser's developer tools are open, specifically the Sources tab, which is currently selected. The code editor shows the HTML and JavaScript files for the page:

```
<body>
  <h1>Botón con evento</h1>
  <button id="boton">Saluda</button>
</body>

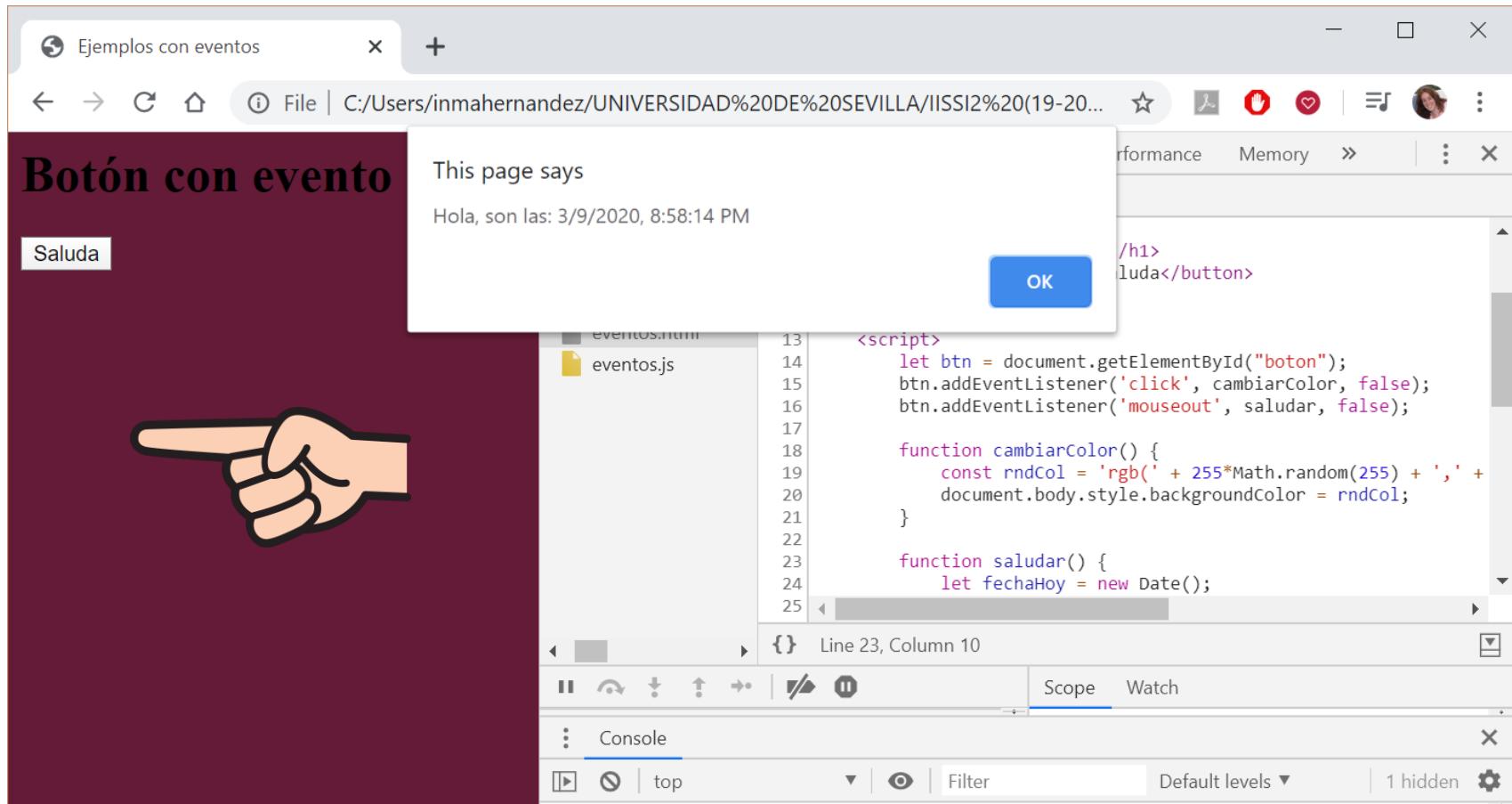
<script>
  let btn = document.getElementById("boton");
  btn.addEventListener('click', cambiarColor, false);
  btn.addEventListener('mouseout', saludar, false);

  function cambiarColor() {
    const rndCol = 'rgb(' + 255*Math.random(255) + ', ' +
      document.body.style.backgroundColor = rndCol;
  }

  function saludar() {
    let fechaHoy = new Date();
  }
</script>
```

The code includes event listeners for 'click' and 'mouseout' events on the button, and two functions: 'cambiarColor' and 'saludar'. The developer tools also show the console tab at the bottom.

# Event Listeners para distintos tipos de evento



The screenshot shows a browser window titled "Ejemplos con eventos". The main content area displays the text "Botón con evento" and "This page says Hola, son las: 3/9/2020, 8:58:14 PM". A modal dialog box is open, containing the text "Saluda" and an "OK" button. On the left side of the browser, there is a large graphic of a hand pointing towards the right. The developer tools are open, showing the "eventos.js" file with the following code:

```
13 <script>
14   let btn = document.getElementById("boton");
15   btn.addEventListener('click', cambiarColor, false);
16   btn.addEventListener('mouseout', saludar, false);
17
18   function cambiarColor() {
19     const rndCol = 'rgb(' + 255*Math.random(255) + ',' +
20       document.body.style.backgroundColor = rndCol;
21   }
22
23   function saludar() {
24     let fechaHoy = new Date();
25 }
```

The code defines a script block with two event listeners: one for 'click' which changes the background color of the body to a random RGB value, and another for 'mouseout' which displays a modal with the text "Saluda". The developer tools also show the DOM structure with an 

# element containing "Hola" and a element.

## Otros métodos

---

- ◆ `removeEventListener()`: elimina un manejador del elemento
- ◆ Forma de uso:  
`elemento.removeEventListener("evento", función)`
- ◆ Efecto: La función deja de estar asociada al manejador del evento (la próxima vez que se produzca el evento ya no se ejecutará).

# Eliminar manejadores - Ejemplo

The screenshot shows a browser window titled "Ejemplos con eventos". The main content area displays a green header with the text "Botón con evento" and a white button labeled "Saluda". The browser's developer tools are open, specifically the "Sources" tab, which is displaying the file "eventos.html". The code in the file is as follows:

```
let btn = document.getElementById("boton");
btn.addEventListener('click', cambiarColor, false);
btn.addEventListener('mouseout', saludar, false);
function cambiarColor() {
  const rndCol = 'rgb(' + 255*Math.random(255) + ',' + 255*Math.random(255) + ',' + 255*Math.random(255) + ')';
  document.body.style.backgroundColor = rndCol;
  btn.removeEventListener('mouseout', saludar);
}
function saludar() {
  let fechaHoy = new Date();
  alert('Hola, son las: ' + fechaHoy.toLocaleString());
}
```

The "Elements" tab shows the DOM structure of the page, including the button element with the id "boton". The "Console" tab is also visible at the top of the developer tools.

# Eliminar manejadores - Ejemplo

The screenshot shows a browser window titled "Ejemplos con eventos". Inside, there's a green box containing the text "Botón con evento" and a button labeled "Saluda". The browser's developer tools are open, specifically the Elements and Sources tabs. The Sources tab shows the file "eventos.html" with the following code:

```
let btn = document.getElementById("boton");
btn.addEventListener('click', cambiarColor, false);
btn.addEventListener('mouseout', saludar, false);
function cambiarColor() {
  const rndCol = 'rgb(' + 255*Math.random(255) + ',' + 255*Math.random(255) + ',' + 255*Math.random(255) + ')';
  document.body.style.backgroundColor = rndCol;
  btn.removeEventListener('mouseout', saludar);
}
function saludar() {
  let fechaHoy = new Date();
  alert('Hola, son las: ' + fechaHoy.toLocaleString());
}
```

The Elements tab shows the DOM structure:

```
<body>
  <h1>Botón con evento</h1>
  <button id="boton">Saluda</button>
</body>
```

A yellow callout box on the right contains the text: "Al quitar el ratón de encima del botón saludará hasta la primera vez que el usuario pulse el botón".

# Eventos load/unload

---

- ◆ Eventos manejados muy frecuentemente en asociación con el elemento body
- ◆ Se disparan cuando el usuario carga completamente la página (incluyendo estilos, scripts, etc) o sale de ella, respectivamente.
  - ◆ Por ejemplo, se puede asociar un manejador al evento load para comprobar la versión del navegador del visitante, y de esta forma cargar una versión de la página optimizada para esa versión
- ◆ También se pueden usar para gestionar las cookies

# Evento onload - Ejemplo

```
<body onLoad="checkBrowser()">
<div id="browserInfo"></div>
```

The screenshot shows a browser window titled "Ejemplos con eventos". The page content displays "Información sobre el navegador" followed by various browser information. At the top, the browser's developer tools are open, specifically the "Sources" tab, which shows the source code for "eventos.html". The code defines a "checkBrowser()" function that concatenates strings containing browser details and sets the result to the innerHTML of a div with id "browserInfo".

Browser CodeName: Mozilla

Browser Name: Netscape

Browser Version: 5.0 (Windows NT 10.0; AppleWebKit/537.36 (KHTML, like Gecko); Chrome/80.0.3987.132 Safari/537.36)

Cookies Enabled: true

Platform: Win32

User-agent header: Mozilla/5.0 (Windows NT 10.0; AppleWebKit/537.36 (KHTML, like Gecko); Chrome/80.0.3987.132 Safari/537.36)

```
function checkBrowser() {
    let txt = "<p>Browser CodeName: "
        + navigator.appCodeName + "</p>";
    txt+= "<p>Browser Name: "
        + navigator.appName + "</p>";
    txt+= "<p>Browser Version: "
        + navigator.appVersion + "</p>";
    txt+= "<p>Cookies Enabled: "
        + navigator.cookieEnabled + "</p>";
    txt+= "<p>Platform: "
        + navigator.platform + "</p>";
    txt+= "<p>User-agent header: "
        + navigator.userAgent + "</p>";
    document.getElementById("browserInfo").innerHTML = txt;
}
```

# Evento unload - Ejemplo

```
<body onUnload="despedida()">
```

The screenshot shows a browser window titled "Ejemplos con eventos". The address bar shows the file path: C:/Users/inmahernandez/UNIVERSIDAD%20DE%20SEVILLA/ISSI2%20(19-20)%20... . The developer tools are open, specifically the Sources tab, displaying the code for "eventos.html". A yellow callout box on the left contains the text: "Ojo: este código no funciona en Chrome". The code in the Sources tab is:

```
function despedida() {
    let nombre = prompt("¿Cómo te llamas?", "");
    alert("¡Adiós, " + nombre + "!");
}
```

In the Sources tab, line 53 is highlighted. The developer tools also show the Call Stack and Scope sections.

# Evento change

---

- ◆ Evento que se suele manejar frecuentemente asociado a los campos de los formularios (sobre todo para la validación de los datos introducidos)

# Evento change - Ejemplo

```
<body>
  <h1>Paso a mayúsculas</h1>
  <input type="text" id="nombre" onchange="toUpperCase()">
</body>
```

## Paso a mayúsculas

The screenshot shows the browser's developer tools open to the 'Sources' tab. The left sidebar lists files: top, file://, C:/Users/inmahernan, eventos.html (which is selected), and eventos.js. The right pane displays the contents of the eventos.js file:

```
37 + navigator.appName + "</p>";
38 " + navigator.appVersion + "</p>";
39 " + navigator.cookieEnabled + "</p>";
40 navigator.platform + "</p>";
41 er: " + navigator.userAgent + "</p>";
42 browserInfo").innerHTML=txt;
43
44
45
46 entById("nombre").value;
47 nombre").value=txt.toUpperCase();
48
49
50
51
52
```

```
function uppercase() {
  let txt = document.getElementById("nombre").value;
  document.getElementById("nombre").value=txt.toUpperCase();
}
```

# Evento change - Ejemplo

```
<body>
  <h1>Paso a mayúsculas</h1>
  <input type="text" id="nombre" onchange="toUpperCase()">
</body>
```

## Paso a mayúsculas

iissi2



The screenshot shows the browser's developer tools open. The 'Sources' tab is selected, displaying the file structure under 'C:/Users/inmahernan'. The 'eventos.html' file is selected, showing its content in the main pane. The code in the file is:

```
37 + navigator.appName + "</p>";
38 + " + navigator.appVersion + "</p>";
39 " + navigator.cookieEnabled + "</p>";
40 navigator.platform + "</p>";
41 er: " + navigator.userAgent + "</p>";
42 browserInfo").innerHTML=txt;
43
44
45
46 entById("nombre").value;
47 nombre").value=txt.toUpperCase();
48
49
50
51
52
```

```
function uppercase() {
  let txt = document.getElementById("nombre").value;
  document.getElementById("nombre").value=txt.toUpperCase();
}
```

# Evento change - Ejemplo

```
<body>
  <h1>Paso a mayúsculas</h1>
  <input type="text" id="nombre" onchange="toUpperCase()">
</body>
```

## Paso a mayúsculas

IISSI2



El evento se  
dispara cuando se  
quita el cursor del  
campo

```
function uppercase() {
  let input = document.getElementById("nombre");
  let txt = input.value;
  input.value = txt.toUpperCase();
}
```

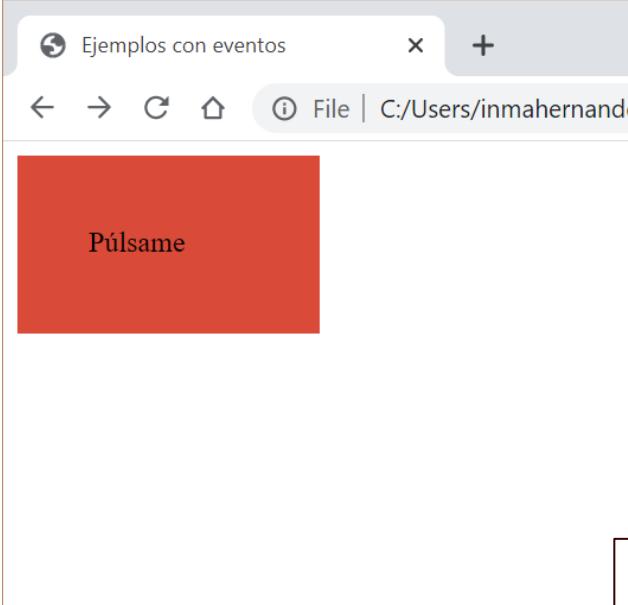
A screenshot of a browser's developer tools showing the "Sources" tab. It displays the code for "eventos.html" and "eventos.js". The "eventos.html" file contains the HTML code shown at the top of the slide. The "eventos.js" file contains the JavaScript function "toUpperCase" defined in the code block above. The code in "eventos.js" uses the "document.getElementById" method to get the input element by its ID, retrieves its current value, converts it to uppercase using the "toUpperCase" method, and then sets the input's value to the uppercase version.

# Eventos de ratón

---

- ◆ Se disparan cuando el usuario interactúa con el ratón con un elemento
  - ◆ mouseover: cuando el ratón está pasando por encima del elemento
  - ◆ mouseout: cuando el ratón deja de pasar por encima del elemento
  - ◆ mousedown: Cuando se pulsa el botón del ratón sobre el elemento
  - ◆ mouseup: Cuando se libera el botón del ratón tras pulsar sobre un elemento
  - ◆ click: cuando se completa la pulsación (mousedown + mouseup)

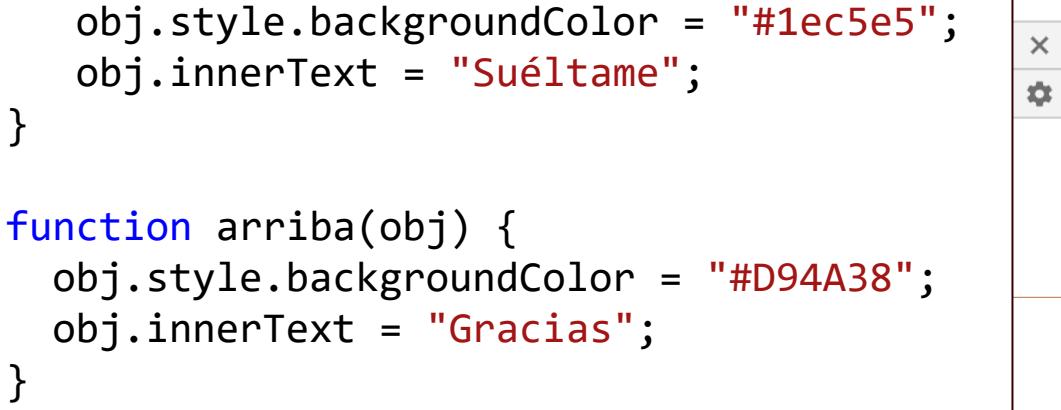
# Eventos de ratón - Ejemplo



The screenshot shows a browser window titled "Ejemplos con eventos". Inside the window, there is a single red button with the text "Púlsame" on it. The browser's address bar shows the path "C:/Users/inmahernandez".

```
<body>
  <div onmousedown="abajo(this)"
        onmouseup="arriba(this)"
        style="background-
color:#D94A38; width:90px; height:20px;
padding:40px;">
    Púlsame</div>
</body>
```

The code above the browser window is the HTML for the button. It includes event handlers for mousedown and mouseup events, both pointing to functions named "abajo" and "arriba" respectively.



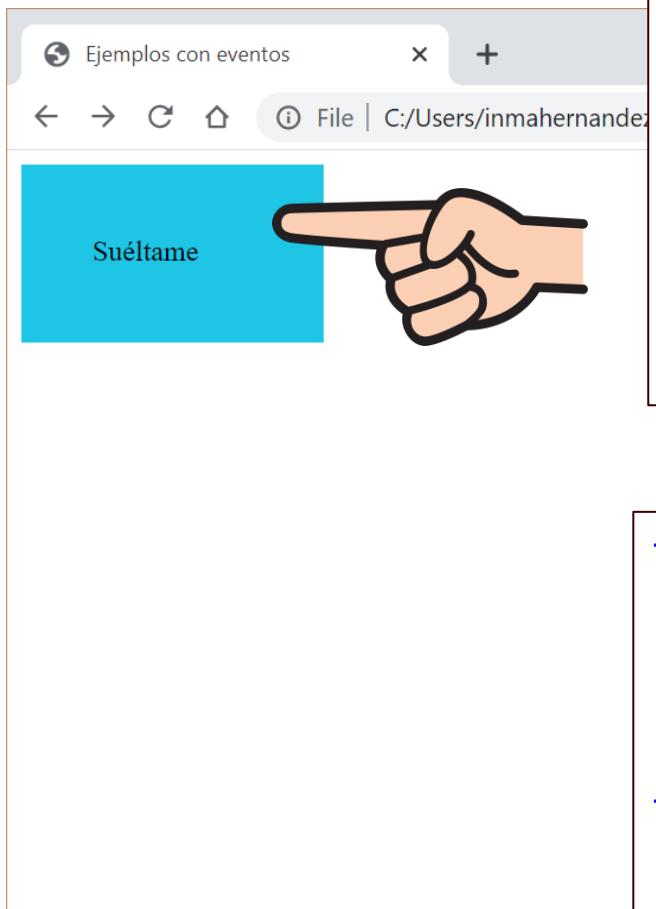
```
function abajo(obj) {
  obj.style.backgroundColor = "#1ec5e5";
  obj.innerText = "Suéltame";
}

function arriba(obj) {
  obj.style.backgroundColor = "#D94A38";
  obj.innerText = "Gracias";
}
```

The screenshot shows a debugger interface with the "Main" scope selected. Two functions are defined: "abajo" and "arriba". The "abajo" function changes the background color of the element to "#1ec5e5" and sets its inner text to "Suéltame". The "arriba" function changes the background color back to "#D94A38" and sets its inner text to "Gracias".

**Se puede usar "this" para pasar el propio elemento HTML**

# Eventos de ratón - Ejemplo

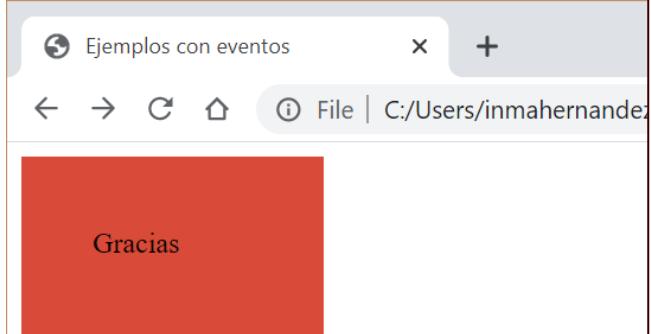


```
<body>
  <div onmousedown="abajo(this)"
        onmouseup="arriba(this)"
        style="background-
color:#D94A38; width:90px; height:20px;
padding:40px;">
    Púlsame</div>
</body>
```

```
function abajo(obj) {
  obj.style.backgroundColor = "#1ec5e5";
  obj.innerHTML = "Suéltame";
}

function arriba(obj) {
  obj.style.backgroundColor = "#D94A38";
  obj.innerHTML = "Gracias";
}
```

# Eventos de ratón - Ejemplo



```
<body>
  <div onmousedown="abajo(this)"
        onmouseup="arriba(this)"
        style="background-
color:#D94A38; width:90px; height:20px;
padding:40px;">
    Púlsame</div>
</body>
```

```
function abajo(obj) {
  obj.style.backgroundColor = "#1ec5e5";
  obj.innerHTML = "Suéltame";
}

function arriba(obj) {
  obj.style.backgroundColor = "#D94A38";
  obj.innerHTML = "Gracias";
}
```

## Evento focus/blur

---

- ◆ Se producen cuando el elemento adquiere/pierde el foco
- ◆ Se suelen emplear para aplicar algún tipo de procesado sobre los campos de los formularios conforme el usuario los va completando

# Eventos focus/blur - Ejemplos

The screenshot shows a browser window titled "Ejemplos con eventos". Inside the window, there is an input field with the ID "nombre". To the right of the browser, the corresponding HTML and JavaScript code are displayed.

```
<body>
  <input type="text" id="nombre">
</body>
```

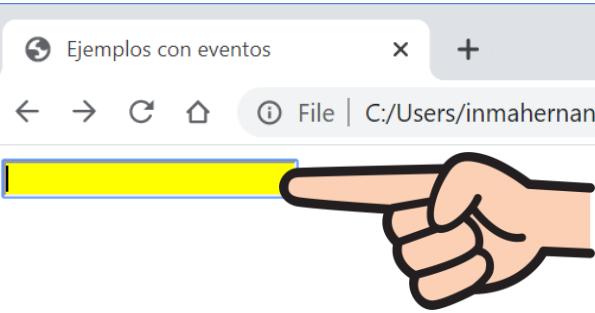
```
function pulsa(obj) {
  alert('click completado');
}
*/
```

```
let input = document.getElementById("nombre");

function foco() {
  input.style.background = "yellow";
}
function fueraFoco() {
  input.style.background = "green";
}

input.onfocus = foco;
input.onblur = fueraFoco;
```

# Eventos focus/blur - Ejemplos



```
<body>
  <input type="text" id="nombre">
</body>
```

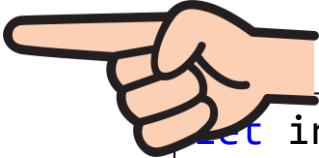
```
let input = document.getElementById("nombre");

function foco() {
  input.style.background = "yellow";
}
function fueraFoco() {
  input.style.background = "green";
}

input.onfocus = foco;
input.onblur = fueraFoco;
```

```
function pulsar(obj) {
  alert('click completado');
}
```

# Eventos focus/blur - Ejemplos



Ejemplos con eventos

File | C:/Users/inmahernandez/

```
<body>
  <input type="text" id="nombre">
</body>
```

file:// C:/Users/inmahernandez/ eventos.html eventos.js

```
function pulsa(obj) {
  alert('click completado');
}
*/
let input = document.getElementById("nombre");

function foco() {
  input.style.background = "yellow";
}
function fueraFoco() {
  input.style.background = "green";
}

input.onfocus = foco;
input.onblur = fueraFoco;
```

# Ejemplo: lista de la compra

```
<body>
  Nuevo elemento: <input type="text" id="input-list">
  <br>
  <button id="btn-clear">Vaciar la lista</button>
  <h1>Lista de la compra</h1>
  <ul id="list"></ul>
</body>
```

Nuevo elemento:

Vaciar la lista

**Lista de la compra**

# Ejemplo: lista de la compra

```
<body>
  Nuevo elemento: <input type="text" id="input-list">
  <br>
  <button id="btn-clear">Vaciar la lista</button>
  <h1>Lista de la compra</h1>
  <ul id="list"></ul>
</body>
```

```
let input = document.getElementById("input-list");
let list = document.getElementById("list");
let clearBtn = document.getElementById("btn-clear");
```

The screenshot shows a user interface for a shopping list. On the left, there is a text input field labeled "Nuevo elemento:" and a button labeled "Vaciar la lista". On the right, the title "Lista de la compra" is displayed above an empty list of items.

# Ejemplo: lista de la compra

```
<body>
  Nuevo elemento: <input type="text" id="input-list">
  <br>
  <button id="btn-clear">Vaciar la lista</button>
  <h1>Lista de la compra</h1>
  <ul id="list"></ul>
</body>
```

```
let input = document.getElementById("input-list");
let list = document.getElementById("list");
let clearBtn = document.getElementById("btn-clear");

function addElement() {
  let element = input.value;
  input.value = "";

  let newItem = document.createElement("li");
  newItem.innerText = element;
  list.appendChild(newItem);
}

input.addEventListener("change", addElement);
```

The screenshot shows a simple web page titled "Lista de la compra". On the left, there is a text input field labeled "Nuevo elemento:" and a button labeled "Vaciar la lista". On the right, there is a heading "Lista de la compra" followed by an empty list.

# Ejemplo: lista de la compra

```
<body>
  Nuevo elemento: <input type="text" id="input-list">
  <br>
  <button id="btn-clear">Vaciar la lista</button>
  <h1>Lista de la compra</h1>
  <ul id="list"></ul>
</body>
```

```
let input = document.getElementById("input-list");
let list = document.getElementById("list");
let clearBtn = document.getElementById("btn-clear");

function addElement() {
  let element = input.value;
  input.value = "";

  let newItem = document.createElement("li");
  newItem.innerText = element;
  list.appendChild(newItem);
}

input.addEventListener("change", addElement);
```

Nuevo elemento:

Vaciar la lista

## Lista de la compra

# Ejemplo: lista de la compra

```
<body>
  Nuevo elemento: <input type="text" id="input-list">
  <br>
  <button id="btn-clear">Vaciar la lista</button>
  <h1>Lista de la compra</h1>
  <ul id="list"></ul>
</body>
```

```
let input = document.getElementById("input-list");
let list = document.getElementById("list");
let clearBtn = document.getElementById("btn-clear");

function addElement() {
  let element = input.value;
  input.value = "";

  let newItem = document.createElement("li");
  newItem.innerText = element;
  list.appendChild(newItem);
}

input.addEventListener("change", addElement);
```



# Ejemplo: lista de la compra

```
<body>
    Nuevo elemento: <input type="text" id="input-list">
    <br>
    <button id="btn-clear">Vaciar la lista</button>
    <h1>Lista de la compra</h1>
    <ul id="list"></ul>
</body>
```

```
let input = document.getElementById("input-list");
let list = document.getElementById("list");
let clearBtn = document.getElementById("btn-clear");

function clearList() {
    let msg = "Are you sure you want to
              clear the list?";
    if(confirm(msg)) {
        list.innerHTML = "";
    }
}

clearBtn.addEventListener("click", clearList);
```

The screenshot shows a user interface for a shopping list. On the left, there is a text input field labeled "Nuevo elemento:" and a button labeled "Vaciar la lista". To the right, the title "Lista de la compra" is displayed in a large, bold font. Below the title, a bulleted list contains the item "• manzanas".

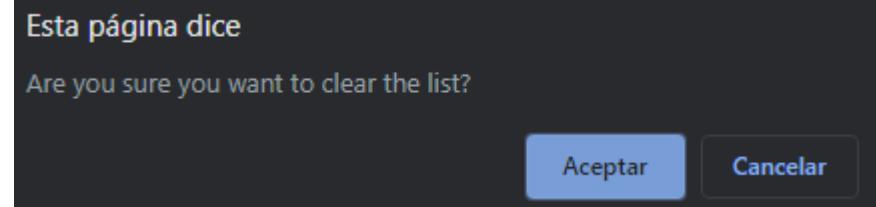
# Ejemplo: lista de la compra

```
<body>
  Nuevo elemento: <input type="text" id="input-list">
  <br>
  <button id="btn-clear">Vaciar la lista</button>
  <h1>Lista de la compra</h1>
  <ul id="list"></ul>
</body>
```

```
let input = document.getElementById("input-list");
let list = document.getElementById("list");
let clearBtn = document.getElementById("btn-clear");

function clearList() {
  let msg = "Are you sure you want to
            clear the list?";
  if(confirm(msg)) {
    list.innerHTML = "";
  }
}

clearBtn.addEventListener("click", clearList);
```



## Otros eventos interesantes

---

- ◆ submit (objeto Form)
- ◆ scroll (objeto Document)
- ◆ keydown/keypress/keyup

# Contenidos

---

1. Introducción
2. Manejo de eventos en JavaScript
3. Ejemplos
- 4. El objeto Event**
5. Validación de formularios

# El objeto Event

---

- ◆ Cuando el listener notifica que se ha producido un evento y se ejecuta el código de una función para manejar dicho evento, se pasa a dicha función como parámetro un objeto que representa el evento en sí
- ◆ Este objeto tiene ciertas propiedades que pueden ser consultadas desde la función
- ◆ Todo objeto evento hereda de la clase Event, pero las propiedades concretas dependen del subtipo de evento de que se trate
  - ◆ Todos tienen la propiedad "target", que devuelve una referencia al elemento HTML que provoca el evento

# Objeto Event - Ejemplo

The screenshot shows a browser developer tools console with the title "Ejemplos con eventos". On the left, there is a sidebar with the following information:

- Target: derft
- Tipo de evento: keydown
- keyCode: 18
- shiftKey: false
- altKey: true
- ctrlKey: false

The main area displays the HTML code for a body element and the corresponding JavaScript code:

```
<body>
  <input type="text" id="nombre">
  <div id="eventInfo"></div>
</body>

let nombreTxt = document.getElementById("nombre");

function verEvento(event) {
  txt = "<p>Tipo de evento: " + event.type + "</p>";
  txt+= "<p>Target: " + event.target.value + "</p>";
  txt+= "<p>keyCode: " + event.keyCode + "</p>";
  txt+= "<p>shiftKey: " + event.shiftKey + "</p>";
  txt+= "<p>altKey: " + event.altKey + "</p>";
  txt+= "<p>ctrlKey: " + event.ctrlKey + "</p>";
  document.getElementById("eventInfo").innerHTML=txt;
}

nombreTxt.addEventListener('keydown', verEvento, false);
```

The file path shown in the code editor is C:/Users/inmahernanc/80 \*/ eventos.html

# Propiedades de los eventos de teclado

Property/Method	Description
<u>altKey</u>	Returns whether the "ALT" key was pressed when the key event was triggered
<u>charCode</u>	Returns the Unicode character code of the key that triggered the event
<u>code</u>	Returns the code of the key that triggered the event
<u>ctrlKey</u>	Returns whether the "CTRL" key was pressed when the key event was triggered
<u>getModifierState()</u>	Returns true if the specified key is activated
<u>isComposing</u>	Returns whether the state of the event is composing or not
<u>key</u>	Returns the key value of the key represented by the event
<u>keyCode</u>	Returns the Unicode character code of the key that triggered the onkeypress event, or the Unicode key code of the key that triggered the onkeydown or onkeyup event
<u>location</u>	Returns the location of a key on the keyboard or device
<u>metaKey</u>	Returns whether the "meta" key was pressed when the key event was triggered
<u>repeat</u>	Returns whether a key is being held down repeatedly, or not
<u>shiftKey</u>	Returns whether the "SHIFT" key was pressed when the key event was triggered
<u>which</u>	Returns the Unicode character code of the key that triggered the onkeypress event, or the Unicode key code of the key that triggered the onkeydown or onkeyup event

[https://www.w3schools.com/jsref/obj\\_keyboardevent.asp](https://www.w3schools.com/jsref/obj_keyboardevent.asp)

# Propiedades de los eventos de ratón

Property/Method	Description
<u>altKey</u>	Returns whether the "ALT" key was pressed when the mouse event was triggered
<u>button</u>	Returns which mouse button was pressed when the mouse event was triggered
<u>buttons</u>	Returns which mouse buttons were pressed when the mouse event was triggered
<u>clientX</u>	Returns the horizontal coordinate of the mouse pointer, relative to the current window, when the mouse event was triggered
<u>clientY</u>	Returns the vertical coordinate of the mouse pointer, relative to the current window, when the mouse event was triggered
<u>ctrlKey</u>	Returns whether the "CTRL" key was pressed when the mouse event was triggered
<u>getModifierState()</u>	Returns true if the specified key is activated
<u>metaKey</u>	Returns whether the "META" key was pressed when an event was triggered
<u>movementX</u>	Returns the horizontal coordinate of the mouse pointer relative to the position of the last mousemove event
<u>movementY</u>	Returns the vertical coordinate of the mouse pointer relative to the position of the last mousemove event
<u>offsetX</u>	Returns the horizontal coordinate of the mouse pointer relative to the position of the edge of the target element
<u>offsetY</u>	Returns the vertical coordinate of the mouse pointer relative to the position of the edge of the target element
<u>pageX</u>	Returns the horizontal coordinate of the mouse pointer, relative to the document, when the mouse event was triggered
<u>pageY</u>	Returns the vertical coordinate of the mouse pointer, relative to the document, when the mouse event was triggered
<u>region</u>	
<u>relatedTarget</u>	Returns the element related to the element that triggered the mouse event
<u>screenX</u>	Returns the horizontal coordinate of the mouse pointer, relative to the screen, when an event was triggered
<u>screenY</u>	Returns the vertical coordinate of the mouse pointer, relative to the screen, when an event was triggered
<u>shiftKey</u>	Returns whether the "SHIFT" key was pressed when an event was triggered
<u>which</u>	Returns which mouse button was pressed when the mouse event was triggered

[https://www.w3schools.com/jsref/obj\\_mouseevent.asp](https://www.w3schools.com/jsref/obj_mouseevent.asp)

# Propiedades de los eventos de campos de formulario

---

Property/Method	Description
<u>data</u>	Returns the inserted characters
dataTransfer	Returns an object containing information about the inserted/deleted data
getTargetRanges()	Returns an array containing target ranges that will be affected by the insertion/deletion
<u>inputType</u>	Returns the type of the change (i.e "inserting" or "deleting")
isComposing	Returns whether the state of the event is composing or not

[https://www.w3schools.com/jsref/obj\\_inputhead.asp](https://www.w3schools.com/jsref/obj_inputhead.asp)

# Propiedades de los eventos táctiles

Property/Method	Description
<u>altKey</u>	Returns whether the "ALT" key was pressed when the touch event was triggered
<u>changedTouches</u>	Returns a list of all the touch objects whose state changed between the previous touch and this touch
<u>ctrlKey</u>	Returns whether the "CTRL" key was pressed when the touch event was triggered
<u>metaKey</u>	Returns whether the "meta" key was pressed when the touch event was triggered
<u>shiftKey</u>	Returns whether the "SHIFT" key was pressed when the touch event was triggered
<u>targetTouches</u>	Returns a list of all the touch objects that are in contact with the surface and where the touchstart event occurred on the same target element as the current target element
<u>touches</u>	Returns a list of all the touch objects that are currently in contact with the surface

[https://www.w3schools.com/jsref/obj\\_touchevent.asp](https://www.w3schools.com/jsref/obj_touchevent.asp)

# Contenidos

---

1. Introducción
2. Manejo de eventos en JavaScript
3. Ejemplos
4. El objeto Event
- 5. Validación de formularios**

# ¿Por qué validar formularios en cliente?

---

- ◆ Permite mostrar al usuario mensajes de error intuitivos antes de enviar el formulario al backend
- ◆ Se puede cancelar el envío del formulario para ahorrar al backend procesar formularios inválidos
- ◆ Podemos decidir enviar el formulario de cualquier otra manera en lugar de la que ofrece por defecto el navegador (por ejemplo, mediante una petición asíncrona AJAX)
- ◆ En cualquier caso, siempre es necesario validar también en backend

# Evento "submit"

---

- ◆ Los elementos <form> emiten un evento "submit" cuando el usuario envía el formulario:

```
<form id="user-form">
    Nombre: <input id="user-firstName" name="firstName" type="text">
    Apellidos: <input id="user-lastName" name="lastName" type="text">
    Edad: <input id="user-age" name="age" type="number" min="18">
    <input type="submit">
</form>
```

```
let form = document.getElementById("user-form");
form.onsubmit = function() {
    alert("Formulario enviado");
}
```

Nombre:

Apellidos:

Edad:

# Evento "submit"

- ◆ Los elementos <form> emiten un evento "submit" cuando el usuario envía el formulario:

```
<form id="user-form">
    Nombre: <input id="user-firstName" name="firstName" type="text">
    Apellidos: <input id="user-lastName" name="lastName" type="text">
    Edad: <input id="user-age" name="age" type="number" min="18">
    <input type="submit">
</form>
```

```
let form = document.getElementById("user-form");
form.onsubmit = function() {
    alert("Formulario enviado");
}
```

Esta página dice  
Formulario enviado

Aceptar

Apellidos:

Edad:



# Evento "submit"

- ♦ Si el manejador del evento devuelve **false**, el envío del formulario se cancela

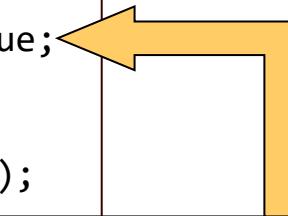
```
<form id="user-form">
    Nombre: <input id="user-firstName" name="firstName" type="text">
    Apellidos: <input id="user-lastName" name="lastName" type="text">
    Edad: <input id="user-age" name="age" type="number" min="18">
    <input type="submit">
</form>
```

```
function validateForm() {
    let valid = true;
    let name = document.getElementById("user-firstName").value;

    if(name.length < 3) {
        alert("The name must be at least 3 characters long!");
        valid = false;
    }

    return valid;
}

form.onsubmit = validateForm;
```



Podemos acceder  
al valor  
introducido en  
cada campo con  
el atributo **value**

# Objetos FormData

- ◆ Los formularios se pueden envolver en objetos FormData para facilitar su consulta:

```
<form id="user-form">
    Nombre: <input id="user-firstName" name="firstName" type="text">
    Apellidos: <input id="user-lastName" name="lastName" type="text">
    Edad: <input id="user-age" name="age" type="number" min="18">
    <input type="submit">
</form>
```

```
let form = document.getElementById("user-form");

function validateForm() {
    let formData = new FormData(form);

    if(formData.get("firstName").length < 3) {
        alert("The name must be at least 3 characters long!");
        res = false;
    }
    return res;
}

form.onsubmit = validateForm;
```

Creamos un nuevo  
FormData a partir  
de la referencia al  
formulario

Los campos se  
referencian por su  
*name*, no por su *id*

# Métodos FormData

---

- ◆ **formData.append("name", "value")**: Añade un nuevo par clave/valor a los datos del formulario
- ◆ **formData.set("name", "value")**: Cambia el valor asociado a una clave, y la añade si ésta no existe
- ◆ **formData.delete("name")**: Elimina una clave (y su valor) de los datos del formulario
- ◆ **formData.get("name")**: Obtiene el valor asociado a una clave
- ◆ **formData.has("name")**: Devuelve un boolean indicando si el formulario tiene asociada una clave indicada
- ◆ **formData.keys()**: Devuelve un iterador sobre todas las claves (atributos *name*) asociadas al formulario
- ◆ **formData.values()**: Devuelve un iterador sobre todos los valores del formulario
- ◆ **formData.entries()**: Devuelve un iterador de todos los pares clave/valor

# Ventajas y desventajas FormData

---

- ◆ Ventajas de usar FormData sobre acceder manualmente a los campos:
  - Permite añadir, eliminar y modificar más fácilmente los valores que se enviarán al servidor
  - Sólo es necesaria una búsqueda en el DOM para obtener el formulario, al contrario que tener que realizar una por cada <input>
  - Se puede enviar directamente el objeto FormData a través de AJAX (lo veremos más adelante)
  - Ofrece una interfaz de programación estándar para acceder y manipular los datos de un formulario: permite el uso de módulos validadores
- ◆ Desventajas:
  - Los valores del objeto FormData son copias de los que hay en los <input>, si se desea modificar el formulario que ve el usuario hay que acceder a ellos y cambiarlos

# Módulos validadores

---

- ◆ Si se envuelve el formulario en un FormData se le puede pasar a otro módulo JS para que lo valide:
  - ◆ Simplifica el código asociado a la gestión de eventos de la página
  - ◆ Permite la combinación mediante composición de validadores

# Módulos validadores

- ◆ Si se envuelve el formulario en un FormData se le puede pasar a otro módulo JS para que lo valide:

```
const userValidator = {           validators/user.js
  validateUser: function(formData) {
    let errors = [];

    if(formData.get("firstName").length < 3) {
      errors.append("The first name is too short");
    }

    if(formData.get("lastName").length < 3) {
      errors.append("The last name is too short");
    }

    if(formData.get("age") < 18) {
      errors.append("You need to be 18+ years old");
    }

    return errors;
  }
};
```

Manejador de submit del form:

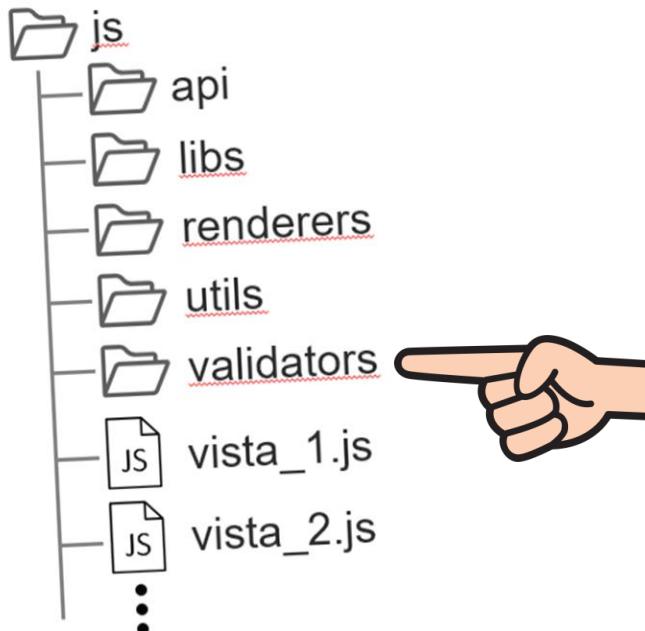
```
import { userValidator } from 'validators/user.js';
let form = document.getElementById("user-form");
let formData = new FormData(form);

let errors = userValidator.validateUser(formData);
if(errors) {
  showErrors(errors);
} else {
  sendForm(formData);
}
```

# Estructura del proyecto

## Organización del código

- ◆ Nosotros seguiremos la siguiente estructura



# Ejercicios

---

- ◆ Implementar una aplicación web que vaya informando de los sitios en los que hace click el ratón
- ◆ Modificar la aplicación anterior para añadir un contador de clicks
- ◆ Modificar la aplicación anterior para contar solamente los clicks que se produzcan en la parte inferior de la pantalla (de la mitad hacia abajo)

# Referencias

---

- ◆ [https://www.w3schools.com/jsref/dom\\_obj\\_event.asp](https://www.w3schools.com/jsref/dom_obj_event.asp)
- ◆ [https://www.w3schools.com/jsref/obj\\_events.asp](https://www.w3schools.com/jsref/obj_events.asp)
- ◆ [https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Building\\_blocks/Events](https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Building_blocks/Events)
- ◆ <https://www.htmlgoodies.com/beyond/javascript/events-and-javascript-part-3-the-event-object.html>



# **Uso de JavaScript (II): Eventos y formularios**

**Departamento de Lenguajes y Sistemas  
Informáticos**  
**Universidad de Sevilla**

*Daniel Ayala, Carlos Arévalo, José Calderón, Margarita Cruz, Inma Hernández, David Ruiz*

**UNIVERSIDAD DE SEVILLA**