

Tarea UI

JavaScript Events



Index

1. Crear una página HTML en la que se van a añadir una serie de eventos asociados a botones.	3
2. Gestionar el comportamiento de un tooltip:.....	5
3. Crear un control deslizante:.....	8
4. Arrastrar objetos por el campo.....	10

1. Crear una página HTML en la que se van a añadir una serie de eventos asociados a botones.

- **Botón mágico:** Se debe crear un botón que se denomine “Objeto mágico” que al darle click sobre el mismo desaparezca.
- **Botón Pulsador:** Se debe crear un botón que se denomine “Pulsador”, donde al pulsarlo debe mostrarse un mensaje “Has pulsado sobre el botón”.
- **Botones deslizantes:** Se deben crear 2 botones que desplieguen y colapsen un listado de opciones, en este caso:
 - **Botón 1:** Tendrá como definición “Películas”, donde se deberá incluir un submenú que al pulsar sobre el botón despliegue/colapse dicho submenú con vuestras 3 series favoritas.
 - **Botón 2:** Tendrá como definición “Series”, donde se deberá incluir un submenú que al pulsar sobre el botón despliegue/colapse dicho submenú con vuestras 3 series favoritas.

HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Exercise 1 - Jose</title>
<style>
.hiddenTheSubmenus {
display: none;
}
</style>
</head>
<body>
<button id="btn-magic">Magic object</button>
<button id="btn-pulsador">Push button</button>

<div>
<button id="btn-movies">Movies</button>
<div id="movies-submenu" class="hiddenTheSubmenus">
<ul>
<li>Morbis</li>
<li>Dragon Ball Bojack</li>
<li>Minecraft: Movie</li>
</ul>
</div>
</div>

<div>
<button id="btn-series">Series</button>
<div id="series-submenu" class="hiddenTheSubmenus">
```

```

<ul>
<li>Inazuma Eleven</li>
<li>Un Show Más</li>
<li>Phineas y Ferb</li>
</ul>
</div>
</div>

<script src="script.js"></script>
</body>
</html>

```

JavaScript:

```

// | Make the magic button disappear | //
document.getElementById('btn-magic').addEventListener('click', function() {
this.style.display = 'none';
});

// | Get a message from push button | //
document.getElementById('btn-pulsador').addEventListener('click', function() {
alert('JSJSJSJSJSJSJ, HAS PULSAO ESTE BOTÓN XDDDDD PRINGAO');
});

// | Get the inside value of movie button and show his display in the class | //
document.getElementById('btn-movies').addEventListener('click', function() {
var submenuOfMovies = document.getElementById('movies-submenu');
if (submenuOfMovies.style.display === 'none' || submenuOfMovies.style.display === '') {
submenuOfMovies.style.display = 'block';
} else {
submenuOfMovies.style.display = 'none';
}
});

// | Get the inside value of serie button and show his display in the class | //
document.getElementById('btn-series').addEventListener('click', function() {
var submenuOfSeries = document.getElementById('series-submenu');
if (submenuOfSeries.style.display === 'none' || submenuOfSeries.style.display === '') {
submenuOfSeries.style.display = 'block';
} else {
submenuOfSeries.style.display = 'none';
}
});

```

2. Gestionar el comportamiento de un tooltip:

Escribe JavaScript que muestre un tooltip sobre un elemento con el atributo data-tooltip.

El valor de este atributo debe convertirse en el texto del tooltip. Solamente un tooltip puede aparecer a la vez.

HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Exercise 2 - Jose</title>
<style>
body {
height: 2000px;
/* El tooltip debe funcionar también después de hacer scroll */
}
.tooltip {
position: fixed;
z-index: 100;
padding: 10px 20px;
border: 1px solid #b3c9ce;
border-radius: 4px;
text-align: center;
font: italic 14px/1.3 sans-serif;
color: #333;
background: #fff;
box-shadow: 3px 3px 3px rgba(0, 0, 0, .3);
display: none;
}
#house {
margin-top: 50px;
width: 400px;
border: 1px solid rgb(70, 29, 29);
}
#roof {
width: 0;
height: 0;
border-left: 200px solid transparent;
border-right: 200px solid transparent;
border-bottom: 20px solid rgb(158, 125, 125);
margin-top: -20px;
}
p {
text-align: justify;
margin: 10px 3px;
}
```

```

</style>
<script src="https://cdn.tailwindcss.com"></script>
</head>
<body>
<div data-tooltip="House" id="house">
<div data-tooltip="Roof" id="roof"></div>
<div class="flex justify-center m-4">
<p><a class="text-red-400 hover:text-red-600 hover:underline"
href="https://youtu.be/GTCB79fDUOY" data-tooltip="Engage the enemy?">Engage the
enemy</a></p>
</div>
</div>

<div class="tooltip" id="tooltip"></div>
<script src="script.js"></script>
</body>
</html>

```

JavaScript:

```

// | Whe get the dom loaded first :x | //
document.addEventListener('DOMContentLoaded', () => {
// | Defined variable tooltip | //
const tooltip = document.getElementById('tooltip');

// | Get the mouseover event | //
document.addEventListener('mouseover', (event) => {
// || Here you get the value of data-tooltip | //
const target = event.target.closest('[data-tooltip]');

// | If the mouse isn't over the element, ends here the code | //
if (!target) return;

// | If the class is data-tooltip the display shows... | //
tooltip.textContent = target.getAttribute('data-tooltip');
tooltip.style.display = 'block';

// | Here you will define where define the position to make appaear the box, if you delete
this, when you rechage the page you must see under the house :x | //
const rect = target.getBoundingClientRect();
const tooltipRect = tooltip.getBoundingClientRect();
tooltip.style.left = rect.left + (rect.width - tooltipRect.width) / 2 + 'px';
tooltip.style.top = rect.top - tooltipRect.height - 5 + 'px';
});

// | Get the mouseout event | //
document.addEventListener('mouseout', (event) => {
// || Here you get the value of data-tooltip | //
const target = event.target.closest('[data-tooltip]');

```

```
// | If the mouse is over the element, ends here the code | //  
if (!target) return;  
// | And you make it invisible | //  
tooltip.style.display = 'none';  
});  
  
// | This is a part of the code for attatch the box with the mouse xD | //  
document.addEventListener('mousemove', (event) => {  
  if (tooltip.style.display === 'block') {  
    // | Get the directions for move the box with the mouse | //  
    tooltip.style.left = event.pageX + 10 + 'px';  
    tooltip.style.top = event.pageY + 10 + 'px';  
  }  
});  
});
```

3. Crear un control deslizante:

A continuación, se muestra un ejemplo de referencia para ver cómo quedaría:



Arrastra el pasador azul con el ratón y muévelo. Detalles importantes:

- Cuando el botón del ratón es presionado, durante el arrastrado del ratón puedes ir por arriba o debajo de la barra deslizante. Ésta seguirá funcionando (es lo conveniente para el usuario).
- Si el ratón se mueve muy rápido hacia la izquierda o la derecha, el pasador se detiene exactamente en el borde

HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Exercise 3 - Jose</title>
<style>
.slider {
border-radius: 5px;
background: #E0E0E0;
background: linear-gradient(left top, #E0E0E0, #EEEEEE);
width: 310px;
height: 15px;
margin: 5px;
}
.thumb {
width: 10px;
height: 25px;
border-radius: 3px;
position: relative;
left: 10px;
top: -5px;
background: rgba(201, 31, 207, 0.781);
cursor: pointer;
}
</style>
</head>
<body>
<div id="slider" class="slider">
<div class="thumb"></div>
</div>
<script src="script.js"></script>
</body>
```


</html>

JavaScript:

```
// | Whe get the dom loaded first | //
document.addEventListener('DOMContentLoaded', () => {
// | Define the classes | //
const slider = document.querySelector('.slider');
const thumb = slider.querySelector('.thumb');

// | We get the thumb | //
thumb.onmousedown = function(event) {
event.preventDefault();

// | We use this for calculate all the movement of the thumb| //
let shiftX = event.clientX - thumb.getBoundingClientRect().left;

// | Get the event of the movement of the mouse and stop to move the thumb with the
mouse | //
document.addEventListener('mousemove', onMouseMove);
document.addEventListener('mouseup', onMouseUp);

// | Here we define a function to the movement of the mouse| //
function onMouseMove(event) {
let newLeft = event.clientX - shiftX - slider.getBoundingClientRect().left;

// | Here we put a limiter so that the x doesn't get out of the slider xD | //
if (newLeft < 0) {
newLeft = 0;
}
let rightEdge = slider.offsetWidth - thumb.offsetWidth;
if (newLeft > rightEdge) {
newLeft = rightEdge;
}

thumb.style.left = newLeft + 'px';
}

// | This is so that we can release the mouse | //
function onMouseUp() {
document.removeEventListener('mousemove', onMouseMove);
document.removeEventListener('mouseup', onMouseUp);
}
};
});
```

4. Arrastrar objetos por el campo

Esta tarea te puede ayudar a comprobar tu entendimiento de varios aspectos de Arrastrar y Soltar, y del DOM. Hacer que todos los elementos con clase draggable sean arrastrables.

- Usa delegación de eventos para detectar el inicio del arrastrado: un solo manejador de eventos en el document para mousedown.
- Si los elementos son arrastrados a los bordes superior/inferior de la ventana: la página se desliza hacia arriba/abajo para permitir dicho arrastre.
- Sin desplazamiento horizontal (esto hace la tarea un poco más simple, añadirlo es fácil).
- Los elementos arrastrables o sus partes nunca deben dejar la ventana, incluso después de movimientos rápidos del ratón.

HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Exercise 4 - Jose</title>
<style>
html, body {
margin: 0;
padding: 0;
}
#field {
background: url(field.svg);
width: 800px;
height: 500px;
float: left;
}

.hero {
background: url(https://js.cx/drag-heroes/heroes.png);
width: 130px;
height: 128px;
float: left;
}

#hero1 {
background-position: 0 0;
}
#hero2 {
background-position: 0 -128px;
}
#hero3 {
background-position: -120px 0;
```

```

}
#hero4 {
background-position: -125px -128px;
}
#hero5 {
background-position: -248px -128px;
}
#hero6 {
background-position: -244px 0;
}
.draggable {
cursor: pointer;
}
</style>
</head>
<body>
<h2>Ubica los superhéroes por el campo.</h2>

<p>Los superhéroes y la pelota son elementos con la clase "draggable". Haz que sean realmente arrastrables.</p>
<p>Importante: limita el arrastre a la ventana. Si el arrastrable alcanza el borde inferior o superior de la ventana, la página debe desplazarse para permitir seguir arrastrando.</p>
<p>Si en tu monitor cabe el documento entero, haz la ventana del navegador más pequeña para que aparezca la barra de desplazamiento vertical y así puedas probarlo.</p>
<p>En esta tarea es suficiente manejar solo el desplazamiento vertical. No suele usarse el desplazamiento horizontal, y el manejo es similar si se necesita.</p>
<p>Y algo más: los héroes nunca pueden dejar la página. Si alcanzan el borde del documento, no pueden arrastrarse fuera de él.</p>
<div id="field">
</div>

<div class="hero draggable" id="hero1"></div>
<div class="hero draggable" id="hero2"></div>
<div class="hero draggable" id="hero3"></div>
<div class="hero draggable" id="hero4"></div>
<div class="hero draggable" id="hero5"></div>
<div class="hero draggable" id="hero6"></div>
<div class="roxas draggable" id="roxas"></div>


<div style="clear:both"></div>
<script src="script.js"></script>
</body>
</html>

```

JavaScript:

```
// | Whe get the dom loaded first | //
```

```

document.addEventListener('DOMContentLoaded', () => {
  // | The variables for make the drag move | //
  let currentDraggable = null;
  let offsetX = 0;
  let offsetY = 0;

  // | Logic of the movement when you want to click and move the img | //
  document.addEventListener('mousedown', (event) => {
    const target = event.target.closest('.draggable');
    if (!target) return;

    currentDraggable = target;
    const rect = target.getBoundingClientRect();

    offsetX = event.clientX - rect.left;
    offsetY = event.clientY - rect.top;

    document.addEventListener('mousemove', onMouseMove);
    document.addEventListener('mouseup', onMouseUp);

    event.preventDefault();
  });

  // | Logic for move the current img | //
  function onMouseMove(event) {
    if (!currentDraggable) return;

    let newLeft = event.clientX - offsetX;
    let newTop = event.clientY - offsetY;

    newLeft = Math.max(0, Math.min(newLeft, window.innerWidth -
currentDraggable.offsetWidth));
    newTop = Math.max(0, Math.min(newTop, window.innerHeight -
currentDraggable.offsetHeight));

    if (newTop > window.innerHeight - 50) {
      window.scrollTo(0, 10);
    } else if (newTop < 50) {
      window.scrollTo(0, -10);
    }

    currentDraggable.style.position = 'absolute';
    currentDraggable.style.left = newLeft + 'px';
    currentDraggable.style.top = newTop + 'px';
  }

  // | Logic for let the mouse out | //
  function onMouseUp() {
    if (!currentDraggable) return;

```

```
document.removeEventListener('mousemove', onMouseMove);
document.removeEventListener('mouseup', onMouseUp);

currentDraggable = null;
}

document.addEventListener('dragstart', (event) => {
event.preventDefault();
});
});
```