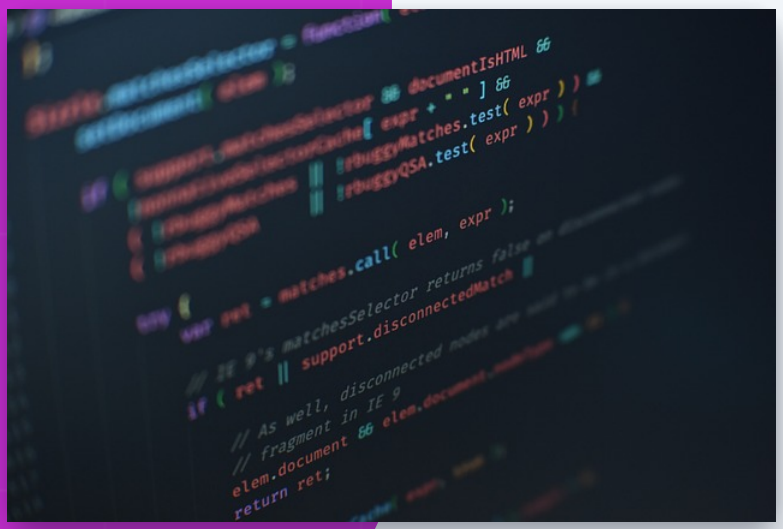

Eventos y manejo de eventos en JavaScript

Aprende a manejar los eventos en JavaScript

Empezar

Descripción general

En este curso aprenderás cómo trabajar con eventos en JavaScript, desde la detección de eventos hasta el manejo de eventos con funciones.

01 Introducción

Introducción a los eventos en JavaScript

01 | Introducción a los eventos en JavaScript

Los eventos son una parte fundamental de la programación en JavaScript, ya que nos permiten interactuar con los elementos de una página web o con el entorno en el que se ejecuta nuestro código. En este tema exploraremos qué son los eventos y cómo podemos utilizarlos en nuestros proyectos.

¿Qué es un evento?

Un evento es una acción que ocurre en el navegador o en el entorno en el que se está ejecutando nuestro código JavaScript. Algunos ejemplos de eventos comunes son hacer clic en un botón, mover el mouse sobre un elemento, presionar una tecla en el teclado o cargar completamente una página web.

Tipos de eventos

En JavaScript, existen diferentes tipos de eventos que podemos utilizar en nuestros proyectos. Algunos de los eventos más comunes son:

- Eventos de ratón: Estos eventos ocurren cuando interactuamos con el mouse. Algunos ejemplos son `click`, `mousedown`, `mouseup`, `mousemove`, entre otros.
- Eventos de teclado: Estos eventos se activan cuando se presionan o sueltan teclas en el teclado. Algunos ejemplos de eventos de teclado son `keydown`, `keyup`, `keypress`.
- Eventos de formulario: Estos eventos se desencadenan cuando se interactúa con un formulario, como enviar o resetear un formulario. Algunos ejemplos son `submit` y `reset`.
- Eventos de carga: Estos eventos ocurren cuando se carga una página web. Algunos ejemplos son `load`, `DOMContentLoaded`.
- Eventos de temporizador: Estos eventos se utilizan para programar acciones que se ejecutan después de un período de tiempo determinado. Algunos ejemplos son `setTimeout`, `setInterval`.

Manejo de eventos

Para capturar y manejar un evento en JavaScript, utilizamos funciones denominadas "manejadores de eventos". Estas funciones se ejecutan

automáticamente cuando ocurre un evento específico.

Podemos añadir un manejador de eventos a un elemento HTML utilizando el atributo `on{evento}`, donde `{evento}` es el nombre del evento que queremos manejar. Por ejemplo, si queremos manejar el evento de hacer clic en un botón, podemos utilizar el atributo `onclick`:

```
<button onclick="miFuncion()">Haz clic</button>
```

En este ejemplo, la función `miFuncion` se ejecutará cuando se haga clic en el botón.

También podemos agregar manejadores de eventos utilizando el método `addEventListener` en JavaScript. Este enfoque nos permite tener más control sobre los eventos y nos permite adjuntar varios manejadores de eventos al mismo elemento. Por ejemplo:

```
 const boton = document.querySelector('button');
boton.addEventListener('click', miFuncion);
```

En este caso, la función `miFuncion` se llamará cada vez que se haga clic en el botón.

Eventos y el modelo de eventos

En JavaScript, los eventos siguen un modelo llamado "modelo de eventos". En este modelo, los eventos se propagan desde el elemento que los desencadena hacia arriba a través de su árbol de elementos padre, esto se conoce como "burbujeo de eventos". Esto significa que si hay elementos anidados dentro de un

elemento y se produce un evento en un elemento interno, también se desencadenará en sus elementos padre.

El modelo de eventos también nos permite detener la propagación de un evento utilizando el método `stopPropagation`. Esto es útil si no queremos que un evento se propague más allá de un cierto elemento.

```
 const boton = document.querySelector('button');
boton.addEventListener('click', function(event) {
  event.stopPropagation();
  // Resto del código
});
```

En este ejemplo, al hacer clic en el botón, el evento no se propagará a otros elementos y será capturado solo por el botón.

Conclusiones

Los eventos en JavaScript nos permiten interactuar con los elementos de una página web y responder a las acciones del usuario. Conocer los diferentes tipos de eventos y cómo manejarlos nos permite crear aplicaciones web interactivas y dinámicas. En futuros temas, exploraremos técnicas más avanzadas de manejo de eventos y cómo utilizar eventos en combinación con otras funcionalidades de JavaScript para crear experiencias de usuario más ricas.

En resumen, el curso de Eventos y manejo de eventos en JavaScript nos introdujo al fascinante mundo de los eventos en JavaScript. Aprendimos cómo funcionan los eventos en el lenguaje y cómo podemos manipularlos para interactuar con la interfaz de usuario. Además, conocimos los diferentes tipos de eventos disponibles, como los eventos de teclado y ratón, y cómo podemos utilizar event listeners para capturar y manejar estos eventos. Con este conocimiento, podemos crear aplicaciones web dinámicas y responsivas, brindando una mejor experiencia al usuario. ¡Esperamos que este curso haya sido de tu agrado y te haya ayudado a mejorar tus habilidades de programación en JavaScript!

Manejo de eventos con Event Listeners

02 | Manejo de eventos con Event Listeners

En JavaScript, los eventos son acciones que suceden en una página web, como hacer clic en un botón, mover el cursor del mouse o cargar una página. El manejo de eventos se refiere a la forma en que podemos detectar y responder a estos eventos en nuestro código JavaScript. Una de las técnicas más comunes para manejar eventos en JavaScript es el uso de **Event Listeners** o "Escuchadores de eventos".

¿Qué es un Event Listener?

Un Event Listener es una función que se adjunta a un elemento HTML y espera a que ocurra un evento específico en ese elemento. Cuando el evento se dispara, es decir, cuando sucede, el Event Listener ejecuta la función asociada. Esto nos permite controlar y responder de manera programática a los eventos que ocurren en una página web.

Agregando un Event Listener

Para agregar un Event Listener a un elemento HTML, usamos el método `addEventListener()`. Este método recibe dos argumentos: el tipo de evento que queremos escuchar y la función que se debe ejecutar cuando el evento ocurra.

```
const button = document.querySelector('#myButton');
```

```
function handleClick(event) {
  console.log('¡Hiciste clic en el botón!');
}

button.addEventListener('click', handleClick);
```

En el ejemplo anterior, seleccionamos un elemento HTML con el identificador `myButton` utilizando el método `querySelector()`. Luego, creamos una función llamada `handleClick` que se ejecutará cuando el evento de clic ocurra en el botón. Finalmente, agregamos el Event Listener usando `addEventListener()` y pasando el tipo de evento ('click') y la función `handleClick`.

Tipos de eventos comunes

Existen muchos tipos de eventos que podemos escuchar y manejar en JavaScript. Algunos de los eventos más comunes incluyen:

- `click`: Se dispara cuando se hace clic en un elemento.
- `mouseover`: Se dispara cuando el cursor del mouse se mueve sobre un elemento.
- `keydown`: Se dispara cuando se presiona una tecla en el teclado.
- `submit`: Se dispara cuando se envía un formulario.

Estos son solo algunos ejemplos de los muchos eventos que se pueden escuchar y manejar en JavaScript.

Removiendo un Event Listener

En algunos casos, es posible que necesitemos eliminar un Event Listener que hemos agregado previamente. Esto se puede hacer utilizando el método `removeEventListener()`. Al igual que `addEventListener()`, `removeEventListener()` también recibe dos argumentos: el tipo de evento y la función que se debe eliminar.

```
 const button = document.querySelector('#myButton');

function handleClick(event) {
  console.log('¡Hiciste clic en el botón!');
}

button.addEventListener('click', handleClick);

// Algunos otros códigos...

button.removeEventListener('click', handleClick);
```

En el ejemplo anterior, agregamos el Event Listener al botón y luego lo eliminamos usando `removeEventListener()`. Esto asegura que la función `handleClick` ya no se ejecutará cuando se haga clic en el botón.

Conclusión - Manejo de eventos con Event Listeners

En conclusión, el tema de la introducción a los eventos en JavaScript nos permitió comprender la importancia de los eventos en la programación web. Aprendimos cómo utilizar la sintaxis de JavaScript para agregar eventos a nuestros elementos HTML y cómo podemos utilizar los eventos para

realizar diferentes acciones cuando el usuario interactúa con nuestra página. Esta introducción sienta las bases para el manejo de eventos en JavaScript y nos brinda una sólida base para aprovechar al máximo las capacidades interactivas de nuestros sitios web.

Eventos del teclado y ratón en JavaScript

Los eventos del teclado y ratón son fundamentales en el desarrollo de aplicaciones interactivas en JavaScript. Estos eventos nos permiten capturar la

interacción del usuario con elementos de la interfaz, como botones, enlaces, formularios y otros elementos HTML.

En este tema, exploraremos en detalle los eventos del teclado y ratón en JavaScript, aprenderemos cómo capturarlos y cómo utilizar la información proporcionada por estos eventos para mejorar la experiencia del usuario.

Eventos del teclado

Los eventos del teclado nos permiten capturar la interacción del usuario con el teclado. Algunos de los eventos más utilizados son:

- `keydown`: Se dispara cuando una tecla es presionada. Podemos obtener información sobre la tecla presionada a través de la propiedad `key` del objeto event.
- `keyup`: Se dispara cuando una tecla es liberada después de haber sido presionada. También podemos obtener información sobre la tecla liberada a través de la propiedad `key` del objeto event.
- `keypress`: Se dispara cuando una tecla es presionada y liberada. Sin embargo, este evento se ha vuelto menos común en la actualidad debido a diferencias de implementación entre navegadores.

Veamos un ejemplo de uso de estos eventos:

```
 document.addEventListener('keydown', function(event) {
  console.log('Tecla presionada:', event.key);
});

document.addEventListener('keyup', function(event) {
  console.log('Tecla liberada:', event.key);
});
```

En este ejemplo, hemos agregado dos listeners de eventos, uno para el evento `keydown` y otro para el evento `keyup`. En el cuerpo de las funciones, simplemente mostramos por consola la tecla presionada o liberada.

Eventos del ratón

Los eventos del ratón nos permiten capturar la interacción del usuario con el ratón o dispositivo de apuntador. Algunos de los eventos más utilizados son:

- `click`: Se dispara cuando el usuario hace clic en un elemento con el botón izquierdo del ratón.
- `dblclick`: Se dispara cuando el usuario hace doble clic en un elemento con el botón izquierdo del ratón.
- `mousedown`: Se dispara cuando se presiona uno de los botones del ratón.
- `mouseup`: Se dispara cuando se libera uno de los botones del ratón después de haber sido presionado.
- `mousemove`: Se dispara cuando el ratón se mueve por encima de un elemento.
- `mouseenter`: Se dispara cuando el ratón entra en un elemento.
- `mouseleave`: Se dispara cuando el ratón sale de un elemento.

Veamos un ejemplo de uso de algunos de estos eventos:

```
 const elemento = document.getElementById('mi-elemento');

elemento.addEventListener('click', function() {
  console.log('Elemento clickeado');
});

elemento.addEventListener('mouseenter', function() {
```

```
  console.log('Ratón sobre el elemento');
});

elemento.addEventListener('mouseleave', function() {
  console.log('Ratón fuera del elemento');
});
```

En este ejemplo, hemos agregado tres listeners de eventos a un elemento con el id 'mi-elemento'. Estos listeners se dispararán cuando se haga clic en el elemento, cuando el ratón entre en el elemento y cuando el ratón salga del elemento, respectivamente. En el cuerpo de las funciones, simplemente mostramos un mensaje por consola.

Conclusion

Los eventos del teclado y ratón son esenciales en el desarrollo de aplicaciones interactivas en JavaScript. Nos permiten capturar la interacción del usuario y utilizar esa información para mejorar la experiencia de usuario. Hemos explorado algunos de los eventos más utilizados, pero hay muchos más eventos disponibles. Conocer estos eventos y cómo utilizarlos nos proporciona herramientas poderosas para crear aplicaciones web dinámicas y atractivas.

Conclusión - Eventos del teclado y ratón en JavaScript

Para concluir, el manejo de eventos con event listeners es una técnica poderosa que nos permite capturar y controlar los eventos que ocurren en nuestros elementos HTML.

Aprendimos cómo agregar event listeners a nuestros elementos y cómo utilizar funciones de callback para manejar los eventos. Esto nos brinda un mayor control sobre la interacción usuario-página y nos permite realizar acciones específicas según los eventos ocurridos. Con esta capacidad, podemos crear experiencias de usuario más dinámicas y atractivas.

Ejercicios Practicos

Pongamos en práctica tus conocimientos

04 | Ejercicios Practicos

En esta lección, pondremos la teoría en práctica a través de actividades prácticas. Haga clic en los elementos a continuación para verificar cada ejercicio y

desarrollar habilidades prácticas que lo ayudarán a tener éxito en el tema.

Manejo de eventos básico

Crea una página web que muestre un botón. Al hacer clic en el botón, se debe mostrar un mensaje en la consola del navegador.

Cambiar el color de un elemento al hacer clic

Crea una página web con un div. Al hacer clic en el div, su color de fondo debe cambiar a un color aleatorio.

Detectar la tecla presionada

Crea una página web que detecte y muestre en la consola del navegador la tecla que ha sido presionada por el usuario.

Resumen

Repasemos lo que acabamos de ver hasta ahora

05 | Resumen

- ✓ En resumen, el curso de Eventos y manejo de eventos en JavaScript nos introdujo al fascinante mundo de los eventos en JavaScript. Aprendimos cómo funcionan los eventos en el lenguaje y cómo podemos manipularlos para interactuar con la interfaz de usuario. Además, conocimos los diferentes tipos de eventos disponibles, como los eventos de teclado y ratón, y cómo podemos utilizar event listeners para capturar y manejar estos eventos. Con este conocimiento, podemos crear aplicaciones web dinámicas y responsivas, brindando una mejor experiencia al usuario. ¡Esperamos que este curso haya sido de tu agrado y te haya ayudado a mejorar tus habilidades de programación en JavaScript!
- ✓ En conclusión, el tema de la introducción a los eventos en JavaScript nos permitió comprender la importancia de los eventos en la programación web. Aprendimos cómo utilizar la sintaxis de JavaScript para agregar eventos a nuestros elementos HTML y cómo podemos utilizar los eventos para realizar

diferentes acciones cuando el usuario interactúa con nuestra página. Esta introducción sienta las bases para el manejo de eventos en JavaScript y nos brinda una sólida base para aprovechar al máximo las capacidades interactivas de nuestros sitios web.

- ✓ Para concluir, el manejo de eventos con event listeners es una técnica poderosa que nos permite capturar y controlar los eventos que ocurren en nuestros elementos HTML. Aprendimos cómo agregar event listeners a nuestros elementos y cómo utilizar funciones de callback para manejar los eventos. Esto nos brinda un mayor control sobre la interacción usuario-página y nos permite realizar acciones específicas según los eventos ocurridos. Con esta capacidad, podemos crear experiencias de usuario más dinámicas y atractivas.
- ✓ En resumen, los eventos del teclado y ratón en JavaScript nos permiten capturar las interacciones del usuario con nuestro sitio web a través de su teclado y ratón. Aprendimos cómo utilizar eventos como 'keydown', 'keyup' y 'click' para realizar acciones específicas cuando el usuario presiona una tecla o hace clic en un elemento. Además, exploramos las propiedades y métodos disponibles para obtener información sobre la tecla o el botón que el usuario interactuó. Con este conocimiento, podemos crear interacciones más ricas y personalizadas en nuestras aplicaciones web.

Prueba

Comprueba tus conocimientos respondiendo unas preguntas

06 | Prueba

Pregunta 1/6

¿Qué es un evento en JavaScript?

- ◯ Una función en JavaScript
- ◯ Un objeto en JavaScript
- ◯ Una acción que ocurre en el navegador

Pregunta 2/6

¿Cómo se puede añadir un event listener en JavaScript?

- ◯ Utilizando el método getEventListener()
- ◯ Utilizando el método addEventListener()
- ◯ Utilizando el método setEventListener()

Pregunta 3/6

¿Cuál es el evento que se dispara cuando se pulsa una tecla en el teclado?

- keydown
- keypress
- keyup

Pregunta 4/6

¿Cuál es el evento que se dispara cuando se hace clic con el ratón?

- click
- mousedown
- mouseup

Pregunta 5/6

¿Cuál es el método utilizado para remover un event listener en JavaScript?

- removeListener()
- removeEventListener()
- deleteListener()

Pregunta 6/6

¿Qué hace el método preventDefault() en un evento?

- Detiene la propagación del evento
- Cancela el evento
- Define el comportamiento predeterminado del evento

Entregar

Conclusión

Felicidades!

¡Felicitaciones por completar este curso! Has dado un paso importante para desbloquear todo tu potencial. Completar este curso no se trata solo de adquirir conocimientos; se trata de poner ese conocimiento en práctica y tener un impacto positivo en el mundo que te rodea.

Comparte este curso