Youtic's
Lic. Gabriel Chaldú

Eventos en JS

Un evento en JavaScript es una acción o ocurrencia que sucede en el navegador web y que puede ser detectada y manejada por el código JavaScript. Estos eventos pueden ser causados por la interacción del usuario con la página web, como hacer clic en un botón, mover el ratón, escribir en un campo de entrada de texto, entre otros, o pueden ser generados por el propio navegador, como la carga completa de una página.

Los eventos permiten que tu código JavaScript responda dinámicamente a las acciones del usuario o a cambios en el estado de la página web. Para capturar y manejar eventos en JavaScript, puedes usar métodos como `addEventListener` para registrar funciones (conocidas como "manejadores de eventos" o "event handlers") que se ejecutarán cuando ocurra el evento especificado.

Aquí hay un ejemplo simple de cómo se puede utilizar `addEventListener` para manejar el evento de clic en un botón HTML:

```
// Obtener una referencia al botón HTML por su ID
const miBoton = document.getElementById('miBoton');
// Agregar un manejador de eventos para el evento 'click'
miBoton.addEventListener('click', function() {
  // Código a ejecutar cuando se hace clic en el botón
  console.log('Se hizo clic en el botón');
});
```

En este ejemplo, cuando el usuario hace clic en el botón con el ID "miBoton", se ejecutará la función proporcionada como manejador de eventos, que simplemente registra un mensaje en la consola.

Los eventos son una parte fundamental de la programación en JavaScript para crear interacciones en tiempo real en tus aplicaciones web, desde formularios interactivos hasta juegos y aplicaciones de usuario más complejas.

Youtic's

Lic. Gabriel Chaldú

DOMContentLoaded

`DOMContentLoaded` es un evento en JavaScript que se dispara cuando el documento HTML ha sido completamente cargado y parseado, lo que significa que la estructura del documento HTML (DOM, Document Object Model) está disponible y lista para ser manipulada a través de JavaScript. Este evento se dispara antes de que se carguen todos los recursos externos, como imágenes y hojas de estilo, lo que lo hace útil para realizar tareas de inicialización que no requieren esperar a que se carguen todos los recursos multimedia.

El evento `DOMContentLoaded` es útil cuando deseas realizar ciertas acciones o vincular eventos a elementos HTML tan pronto como sea posible después de que el documento HTML esté listo, sin tener que esperar a que todos los recursos adicionales se carguen por completo.

Ejemplo de cómo utilizar `DOMContentLoaded`:

```
document.addEventListener('DOMContentLoaded', function() {
    // Este código se ejecutará cuando el documento HTML esté
completamente cargado y listo para ser manipulado

    // Por ejemplo, puedes acceder a elementos HTML y agregar eventos
aquí
    const miBoton = document.getElementById('miBoton');
    miBoton.addEventListener('click', function() {
      alert('¡Se hizo clic en el botón!');
    });
  });
```

En este ejemplo, el código se ejecutará una vez que el documento HTML esté listo, lo que garantiza que el botón con el ID "miBoton" esté disponible para ser manipulado y que el evento de clic se agregue correctamente. Esto ayuda a evitar problemas en los que el código JavaScript intente interactuar con elementos que aún no se han cargado en el DOM.

Los eventos más utilizados en JavaScript se pueden dividir en varias categorías según su origen y propósito. Aquí hay una lista de algunos eventos comunes junto con los elementos HTML en los que se suelen utilizar:

1. Eventos de ratón (Mouse Events):

- `**click**`: Se dispara cuando se hace clic en un elemento.

- `**mouseover**` y `**mouseout**`: Se disparan cuando el cursor del ratón entra y sale de un elemento.

- `**mousedown**`, `**mouseup**`: Se disparan cuando se presiona y se libera un botón del ratón mientras se está sobre un elemento.

2. Eventos de teclado (Keyboard Events):

- `**keydown**` y `**keyup**`: Se disparan cuando se presiona y se suelta una tecla del teclado mientras un elemento tiene el foco, como un campo de entrada de texto.

3. Eventos de formulario (Form Events):

- `**submit**`: Se dispara cuando se envía un formulario.

- `**input**` y `**change**`: Se disparan cuando se cambia el valor de un campo de entrada de texto u otro elemento de formulario.

4. Eventos de ventana (Window Events):

- `**load**`: Se dispara cuando se ha cargado completamente una página web.

- `**resize**`: Se dispara cuando se cambia el tamaño de la ventana del navegador.

- `**scroll**`: Se dispara cuando se desplaza la página.

5. Eventos de tiempo (Time Events):

- `**setTimeout**` y `**setInterval**`: No son eventos en sí, pero se utilizan para ejecutar código después de un retraso o a intervalos regulares.

6. Eventos de arrastrar y soltar (Drag and Drop Events):

- `**dragstart**`, `**dragover**`, `**dragenter**`, `**dragleave**`, `**dragend**`, `**drop**`: Se utilizan en elementos que pueden arrastrarse y soltarse, como imágenes o elementos de lista.

7. Eventos de medios (Media Events):

- `**play**`, `pause`, `**ended**`: Se utilizan en elementos de medios, como elementos de audio y video, para controlar la reproducción.

8. Eventos de enfoque (Focus Events):

- `**focus**` y `**blur**`: Se disparan cuando un elemento obtiene o pierde el foco, como un campo de entrada de texto.

9. Eventos de errores (Error Events):

- `**error**`: Se dispara cuando se produce un error al cargar un recurso, como una imagen o un script.

10. Eventos de historial (History Events):

- `**popstate**`: Se dispara cuando cambia el estado del historial del navegador.

Estos son solo algunos ejemplos de eventos comunes en JavaScript. Puedes adjuntar manejadores de eventos a prácticamente cualquier elemento HTML, como botones, enlaces, divs, imágenes, etc., dependiendo de tus necesidades y de la interacción que desees habilitar en tu página web o aplicación.