**DOCUMENTACIÓN CÓDIGO HTML**



* <!DOCTYPE html>: Esta línea declara el tipo de documento como HTML.
* <html lang="en">: Define el documento como un documento HTML y especifica el idioma principal del contenido como inglés ("en").
* <meta charset="UTF-8">: Define la codificación de caracteres del documento como UTF-8, permitiendo mostrar correctamente caracteres especiales.
* <meta http-equiv="X-UA-Compatible" content="IE=edge">: Esta metaetiqueta es específica para navegadores Internet Explorer. Sirve para indicar compatibilidad con versiones más recientes del motor de renderizado.
* <meta name="viewport" content="width=device-width, initial-scale=1.0">: Esta metaetiqueta controla el ancho y el zoom inicial de la página web. En este caso, se define que el ancho sea igual al ancho del dispositivo y el zoom inicial sea 1.0 (sin zoom).
* <title>Torres de Hanói</title>: Define el título de la página web que se mostrará en la pestaña del navegador.
* <link rel="stylesheet" href="styles.css">: Incluye una hoja de estilos externa llamada "styles.css" para definir la apariencia de la página.
* <link rel="stylesheet" href="assets/fontawesome/css/fontawesome.css">: Incluye la librería Font Awesome para proveer iconos vectoriales. Se incluyen tres archivos relacionados: fontawesome.css, brands.css y solid.css
* <script src="script.js"></script>: Incluye un archivo JavaScript externo llamado "script.js" que contiene el código dinámico de la página.

**Texto

Descripción generada automáticamente con confianza media**

* Elemento body: Define el cuerpo principal de la página web. El atributo id="body" asigna un identificador único a este elemento para que pueda ser referenciado en CSS o JavaScript.
* Elemento header: Define el encabezado de la página web. Contiene un elemento div que muestra el título del juego: "Torres de Hanói".
* Elemento main: Define el contenido principal de la página web. Contiene una sección con las siguientes características:
* Clase screen menu: Esta clase se utiliza para definir el estilo de la sección.
* ID screen: Asigna un identificador único a esta sección para que pueda ser referenciada en CSS o JavaScript.

Dentro de la sección, encontramos un elemento div con la clase infoL que contiene:

* Botón con ID infoL y estilo personalizado: Este botón tiene un icono de información y ejecuta la función MessageInfo() al hacer clic.
* Texto "Instrucciones del juego": Este texto muestra las instrucciones del juego "Torres de Hanói".

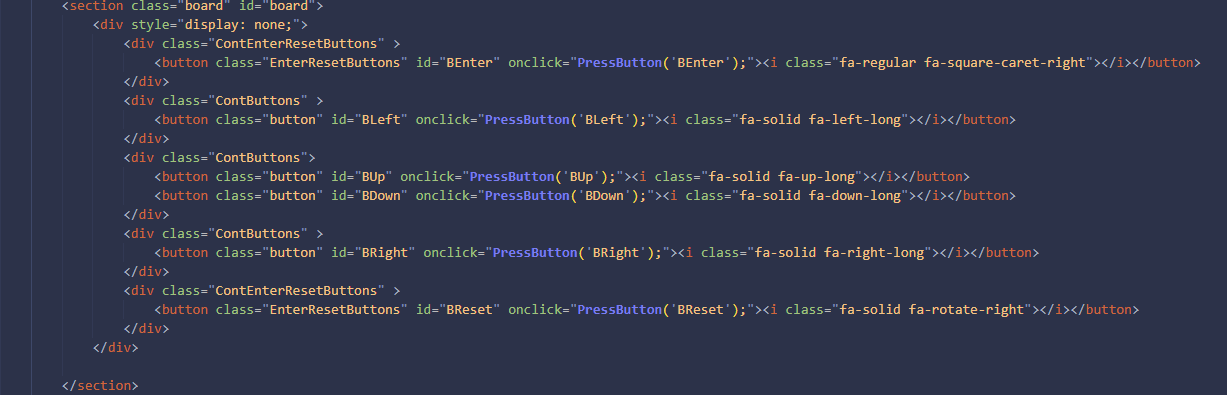
Texto

Descripción generada automáticamente

* Div con clase "OP OPSELECT" e id "OPM0": Este elemento es un botón con el texto "Iniciar".
* Div con clase "OP" e id "OPM1": Este elemento muestra el número de discos seleccionados.
* Etiqueta "label": Esta etiqueta asocia el texto "Nombre del Jugador:" con el siguiente elemento de entrada.
* Atributo "for": El valor de este atributo, "playerName", debe coincidir con el id del elemento de entrada al que se asocia la etiqueta.
* Entrada de Texto “Input type=”text”: La entrada de texto donde se ingresa el nombre del jugador



* <section class="scoreboard" id="scoreboard" style="display: flex;">: Define una sección con la clase "scoreboard", el ID "scoreboard" y un estilo de visualización flexible.
* <h2>Tabla de Puntajes</h2>: Establece un título de nivel 2 para la sección: "Tabla de Puntajes".
* <div class="scores">: Crea un contenedor con la clase "scores" para agrupar las tablas de puntajes.
* <div class="timeScores">: Define una sección para los puntajes por tiempo.
* <h3>Por Tiempo</h3>: Establece un título de nivel 3 para esta sección: "Por Tiempo".
* <table id="timeTable">: Crea una tabla con la ID "timeTable" para mostrar los puntajes por tiempo.
* <thead>: Define la sección del encabezado de la tabla.
* <tr>: Define una fila del encabezado.
* <th>Jugador</th>: Crea una celda de encabezado para la columna "Jugador".
* <th>Tiempo</th>: Crea una celda de encabezado para la columna "Tiempo".
* <tbody>: Define la sección del cuerpo de la tabla donde se mostrarán los datos de los jugadores y sus tiempos.
* <div class="moveScores">: Define una sección para los puntajes por movimientos.
* <h3>Por Movimientos</h3>: Establece un título de nivel 3 para esta sección: "Por Movimientos".
* <table id="movesTable">: Crea una tabla con la ID "movesTable" para mostrar los puntajes por movimientos.
* <thead>: Define la sección del encabezado de la tabla.
* <tr>: Define una fila del encabezado.
* <th>Jugador</th>: Crea una celda de encabezado para la columna "Jugador".
* <th>Movimientos</th>: Crea una celda de encabezado para la columna "Movimientos".
* <tbody>: Define la sección del cuerpo de la tabla donde se mostrarán los datos de los jugadores y sus movimientos.

****

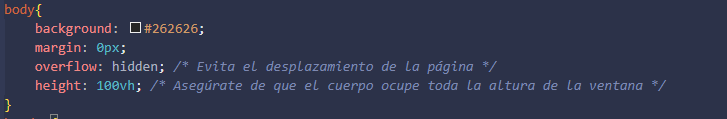
* Sección tablero (<section class="board" id="board">): Define una sección con la clase board y un identificador único id="board".
* Div oculto (<div style="display: none;">): Define un contenedor (div) oculto mediante la propiedad style="display: none;".
* Elementos dentro del div oculto: Varios div con clases que parecen agrupar botones relacionados.
* Botones con clases button y EnterResetButtons que tienen identificadores únicos (id) y llaman a una función PressButton al hacer clic (onclick).
* Íconos de FontAwesome (<i>) representados por clases como fa-regular fa-square-caret-right, fa-solid fa-left-long, etc.

**DOCUMENTACIÓN CÓDIGO CSS**

**Interfaz de usuario gráfica, Texto

Descripción generada automáticamente**

* :root es un pseudo-elemento que selecciona el elemento raíz del documento HTML.
* --colorRest es el nombre de la variable CSS personalizada.
* #40bb34 es el valor de la variable CSS personalizada, que en este caso es un color verde.

****

**Selector body:**

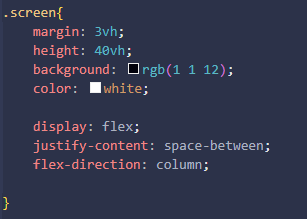
* **background: #262626;**: Esta propiedad establece el color de fondo del cuerpo en gris oscuro (#262626).
* **margin: 0px;**: Esta propiedad elimina los márgenes predeterminados del cuerpo, lo que permite que el contenido se ajuste a los bordes de la ventana.
* **overflow: hidden;**: Esta propiedad evita que aparezcan barras de desplazamiento cuando el contenido del cuerpo excede la altura de la ventana.
* **height: 100vh;**: Esta propiedad establece la altura del cuerpo al 100% de la altura de la ventana del navegador ("vh" significa "viewport height"). Esto asegura que el cuerpo ocupe toda la altura disponible, sin importar el tamaño de la pantalla.

**Texto

Descripción generada automáticamente**

**Selector header:**

* background: #171717;: Esta propiedad establece el color de fondo del encabezado en un gris oscuro (#171717).
* height: 10vh;: Esta propiedad define la altura del encabezado como el 10% de la altura del viewport (la ventana del navegador).
* font-size: 4vh;: Esta propiedad establece el tamaño de la fuente del texto dentro del encabezado en el 4% de la altura del viewport.
* color: #dcdcdc;: Esta propiedad define el color del texto dentro del encabezado en un gris claro (#dcdcdc).
* text-align: center;: Esta propiedad centra el texto horizontalmente dentro del encabezado.
* display: flex;: Esta propiedad cambia el modo de visualización del encabezado a un contenedor flexible.
* justify-content: center;: Esta propiedad centra horizontalmente los elementos dentro del contenedor flexible.
* align-items: center;: Esta propiedad centra verticalmente los elementos dentro del contenedor flexible.

****

**Selector .screen:**

* **margin:** 3vh; Esta propiedad define un margen de 3 unidades de ventana ("vh") alrededor del elemento.
* **height:** 40vh; Esta propiedad establece la altura del elemento en 40 unidades de ventana.
* **background:** rgb(1 1 12); Esta propiedad define el color de fondo del elemento como negro (#000000).
* **color:** white; Esta propiedad establece el color del texto del elemento en blanco (#FFFFFF).
* **display:** flex; Esta propiedad cambia el modo de visualización del elemento a un contenedor flexible.
* **justify-content:** space-between; Esta propiedad define cómo se distribuye el espacio libre entre los elementos secundarios del contenedor flexible.
* **flex-direction:** column; Esta propiedad define la dirección en la que se organizan los elementos secundarios del contenedor flexible.

****

**Selector** **.board:**

* margin-top: 3vh;: Esta propiedad agrega un margen superior de 3 unidades de altura de viewport (vh) al elemento .board.
* height: 40vh;: Esta propiedad establece la altura del elemento .board en 40 unidades de altura de viewport (vh).
* justify-content: space-between;: Esta propiedad define cómo se distribuyen los elementos del contenedor .board a lo largo del eje horizontal.
* flex-wrap: wrap;: Esta propiedad permite que los elementos del contenedor .board se envuelvan en varias filas si no caben en una sola fila.
* align-content: center;: Esta propiedad define cómo se alinean los elementos del contenedor .board a lo largo del eje vertical.

**Texto

Descripción generada automáticamente**

**Selector .ContButtons:**

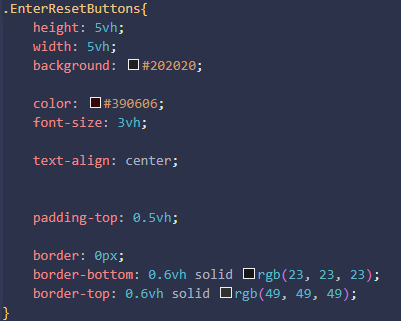
* height: 2vh;: Esta propiedad establece la altura del elemento al 2% de la altura del viewport (la ventana del navegador).
* margin: 1vh;: Esta propiedad añade un margen de 1vh (1% de la altura del viewport) alrededor del elemento.
* height: 100%;: Esta propiedad establece la altura del elemento al 100% de su contenedor padre.
* display: flex;: Esta propiedad cambia el modo de visualización del elemento a flexbox.
* justify-content: center;: Esta propiedad alinea los elementos horizontalmente en el centro del contenedor.
* flex-direction: column;: Esta propiedad establece la dirección en la que se distribuyen los elementos dentro del contenedor.

**Texto

Descripción generada automáticamente**

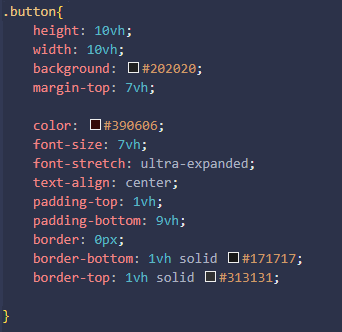
**Selector .ContEnterResetButtons:**

* margin: 0px;: Los valores 0px para margin-top, margin-right, margin-bottom y margin-left se establecen simultáneamente.



**Clase .EnterResetButtons:**

* height (altura): Establece la altura del botón a 5vh (5% del alto del viewport).
* width (ancho): Establece el ancho del botón a 5vh (igual que la altura).
* background (fondo): Define el color de fondo del botón como #202020 (negro).
* color (color): Establece el color del texto del botón como #390606 (un rojo oscuro).
* font-size (tamaño de fuente): Define el tamaño de la fuente del texto del botón a 3vh (basado en el alto del viewport).
* text-align (alineación de texto): Centra el texto del botón horizontalmente.
* padding-top (relleno superior): Agrega un pequeño margen superior de 0.5vh al texto del botón.
* border (borde): Elimina cualquier borde predefinido del botón.
* border-bottom (borde inferior): Añade un borde inferior sólido de 0.6vh de ancho en color rgb(23, 23, 23) (un gris muy oscuro).
* border-top (borde superior): Añade un borde superior sólido de 0.6vh de ancho en color rgb(49, 49, 49) (un gris oscuro).

****

**Elemento .button:**

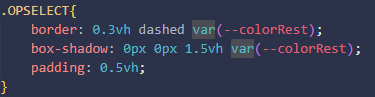
* height: 10vh; - Establece la altura del botón al 10% del alto del viewport (la ventana del navegador).
* width: 10vh; - Define el ancho del botón también al 10% del alto del viewport.
* background: #202020; - Color de fondo del botón, en este caso un gris oscuro (#202020).
* margin-top: 7vh; - Agrega un margen superior de 7vh al botón, separándolo 7% del alto del viewport del elemento anterior.
* color: #390606; - Define el color del texto del botón, un rojo oscuro (#390606).
* font-size: 7vh; - Establece el tamaño de la fuente del texto al 7% del alto del viewport.
* font-stretch: ultra-expanded; - Aplica un estiramiento ultra expandido al texto del botón, haciéndolo más ancho.
* text-align: center; - Alinea el texto del botón horizontalmente al centro.
* padding-top: 1vh; - Agrega un margen interior superior de 1vh al botón.
* padding-bottom: 9vh; - Agrega un margen interior inferior de 9vh al botón.
* border: 0px; - Elimina cualquier borde predefinido del botón.
* border-bottom: 1vh solid #171717; - Agrega un borde inferior de 1vh de grosor con estilo sólido y color gris oscuro (#171717).
* border-top: 1vh solid #313131; - Agrega un borde superior de 1vh de grosor con estilo sólido y color gris oscuro (#313131).

Texto

Descripción generada automáticamente

**Selector .OP:**

* margin: 1vh;: Esta propiedad establece un margen de 1 unidad de altura de viewport (vh) alrededor del elemento seleccionado.
* padding: 0.8vh;: Esta propiedad establece un relleno de 0.8 unidades de altura de viewport dentro del elemento seleccionado.



**Selector .OPSELECT:**

* border: Se establece un borde discontinuo con un grosor de 0.3vh y el color var(--colorRest).
* box-shadow: Esta propiedad agrega una sombra a la caja del elemento. En este caso, se agrega una sombra con un desplazamiento de 0px en ambas direcciones (horizontal y vertical), un desenfoque de 1.5vh y el color var(--colorRest).
* padding: Se agrega un relleno de 0.5vh en todas las direcciones (arriba, abajo, izquierda y derecha).

Texto

Descripción generada automáticamente

**Clase .arrowSpace:**

* width: inherit;: El ancho del elemento heredará el ancho de su contenedor padre.
* height: 10vh;: La altura del elemento será igual al 10% del alto del viewport (la parte visible de la ventana del navegador).
* display: flex;: El elemento se convierte en un contenedor flexible.
* flex-direction: row;: Los elementos secundarios del contenedor se posicionarán uno al lado del otro en una fila horizontal.
* align-content: center;: Si los elementos secundarios no ocupan toda la altura del contenedor flexible, esta propiedad define cómo se alinean verticalmente dentro del contenedor.
* flex-wrap: nowrap;: Si los elementos secundarios sobrepasan el ancho del contenedor flexible, esta propiedad define si se permite el salto de línea para mostrarlos en varias filas. Aquí se establece que no se permite el salto (nowrap), por lo que todos los elementos se mostrarán en una sola fila.

Texto

Descripción generada automáticamente

**Clase .arrow:**

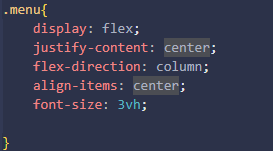
* width: Establece el ancho del elemento al 100% de su contenedor padre.
* height: Define el alto del elemento a 6 unidades de la altura del viewport (vh).
* margin: Agrega un margen de 2vh alrededor del elemento.
* color: Establece el color del texto del elemento a blanco.
* font-size: Define el tamaño de la fuente del texto a 5vh.
* text-align: Centra el texto horizontalmente dentro del elemento.
* display: flex: Configura el elemento para que se comporte como un contenedor flexible.
* justify-content: center: Alinea el contenido horizontalmente en el centro del contenedor flexible.
* flex-direction: column: Define la dirección del flujo de los elementos secundarios como columnas apiladas verticalmente.
* align-items: center: Alinea los elementos secundarios del contenedor flexible en el centro del eje principal (vertical en este caso).

Texto

Descripción generada automáticamente

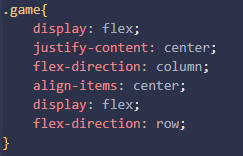
**Selector .timer:**

* font-family: monospace;: Esta propiedad establece la fuente del texto a una fuente monoespaciada.
* color: #00f200;: Esta propiedad establece el color del texto a verde lima (#00f200).
* position: relative;: Esta propiedad establece la posición del elemento como relativa a su posición normal en el flujo del documento. Esto permite que el elemento sea posicionado usando las propiedades top, left, bottom y right.
* left: 0px;: Esta propiedad establece la posición horizontal del elemento a 0 píxeles desde el borde izquierdo de su contenedor.
* align-self: center;: Esta propiedad alinea el elemento horizontalmente al centro de su contenedor.
* font-size: 2vh;: Esta propiedad establece el tamaño de la fuente al 2% de la altura del viewport.



**Selector .menu:**

* display: flex;: Esta propiedad configura el elemento como un contenedor flexible.
* justify-content: center;: Esta propiedad alinea los elementos hijos del contenedor a lo largo del eje principal (horizontal por defecto).
* flex-direction: column;: Esta propiedad define la dirección del eje principal del contenedor flexible.
* align-items: center;: Esta propiedad alinea los elementos hijos del contenedor a lo largo del eje transversal (vertical por defecto).
* font-size: 3vh;: Esta propiedad establece el tamaño de la fuente para todos los elementos dentro del contenedor a 3 unidades de la altura del viewport (ventana gráfica).



**Selector .game:**

* display: flex;: Esta propiedad establece que el elemento contenedor será un contenedor flexible.
* justify-content: center;: Esta propiedad alinea los elementos hijos a lo largo del eje principal (horizontal por defecto) del contenedor flexible, centrándolos en el espacio disponible.
* flex-direction: column;: Esta propiedad establece la dirección en la que se dispondrán los elementos hijos dentro del contenedor flexible.
* align-items: center;: Esta propiedad alinea los elementos hijos a lo largo del eje secundario (vertical por defecto) del contenedor flexible, centrándolos en el espacio disponible.
* flex-direction: row;: Esta propiedad establece la dirección en la que se dispondrán los elementos hijos dentro del contenedor flexible.

Texto

Descripción generada automáticamente

**Selector .contStack:**

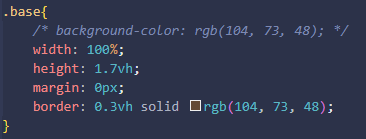
* display: flex: Esta propiedad indica que el elemento contenedor será un contenedor de flexbox.
* justify-content: space-evenly: Esta propiedad distribuye los elementos hijos del contenedor de flexbox de manera uniforme con espacio entre ellos.
* flex-direction: row: Esta propiedad establece que los elementos hijos del contenedor de flexbox se posicionarán en una fila, uno al lado del otro.
* align-items: center: Esta propiedad alinea los elementos hijos del contenedor de flexbox verticalmente en el centro del contenedor.
* height: 90%;: Esta propiedad establece la altura del elemento contenedor al 90% del tamaño de su contenedor padre.
* width: 100%;: Esta propiedad establece el ancho del elemento contenedor al 100% del tamaño de su contenedor padre.
* margin: 2vh: Esta propiedad agrega un margen vertical de 2 unidades de vh (porcentaje de la altura del viewport) al elemento contenedor.

Texto

Descripción generada automáticamente

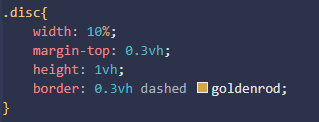
**Selector .Post:**

* height: 90%;: Establece la altura del elemento al 90% del contenedor padre.
* width: 100%;: Establece el ancho del elemento al 100% del contenedor padre.
* margin: 2vh;: Agrega un margen alrededor del elemento equivalente a 2 unidades verticales (vh).
* display: flex;: Activa el modo flexbox para el elemento, permitiendo un control más preciso de la distribución de su contenido.
* align-items: center;: Alinea los elementos secundarios del contenedor flexbox en el centro del eje vertical.
* flex-wrap: wrap;: Permite que los elementos secundarios se envuelvan a la siguiente línea si no caben en una sola fila dentro del contenedor.
* justify-content: flex-end;: Alinea los elementos secundarios del contenedor flexbox en el extremo derecho del eje horizontal.
* flex-direction: column;: Define la dirección del contenedor flexbox como una columna, apilando los elementos secundarios uno encima del otro.



**Selector .base:**

* background-color: (Comentado) Esta propiedad define el color de fondo del elemento. El valor rgb(104, 73, 48) corresponde a un marrón oscuro.
* width: Esta propiedad establece el ancho del elemento al 100% del ancho disponible.
* height: Esta propiedad establece la altura del elemento al 1.7% del alto del viewport (la ventana del navegador).
* margin: Esta propiedad establece los márgenes del elemento a 0px. Esto significa que no habrá espacio alrededor del elemento.
* border: Esta propiedad define el borde del elemento. Se establece un borde sólido de 0.3vh de ancho con color rgb(104, 73, 48) (marrón oscuro).



**Selector .disc:**

* width: 10%: Esta propiedad establece el ancho del elemento al 10% del ancho de su contenedor padre.
* margin-top: 0.3vh: Esta propiedad define un margen superior de 0.3 unidades de viewport height (vh) para el elemento.
* height: 1vh: Esta propiedad establece la altura del elemento a 1 unidad de viewport height (vh).
* border: 0.3vh dashed goldenrod: Esta propiedad define un borde discontinuo de 0.3 unidades de viewport height (vh) de color dorado para el elemento.

Texto

Descripción generada automáticamente

**Selector .endGame:**

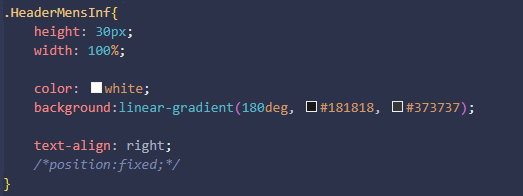
* display: flex;: Cambia el modo de visualización del elemento a flexbox. Esto permite distribuir el contenido del elemento de forma flexible.
* align-content: center;: Alinea el contenido del elemento verticalmente en el centro del contenedor.
* justify-content: center;: Alinea el contenido del elemento horizontalmente en el centro del contenedor.
* flex-wrap: wrap;: Permite que el contenido del elemento se envuelva en varias líneas si no cabe en una sola línea.
* align-items: center;: Alinea los elementos del elemento verticalmente en el centro del contenedor.
* text-align: center;: Alinea el texto del elemento horizontalmente en el centro del contenedor.

Texto

Descripción generada automáticamente

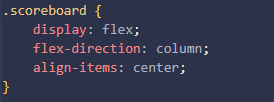
**Selector #MensInf:**

* height: 80vh;: Esta propiedad establece la altura del elemento al 80% del tamaño de la ventana gráfica (viewport height). "vh" es una unidad de medida que representa el 1% de la altura de la ventana.
* width: 80vh;: Similar a la propiedad anterior, esta establece el ancho del elemento al 80% del tamaño de la ventana gráfica.
* background: #101010;: Define el color de fondo del elemento como un gris oscuro (#101010).
* border: 0.2vh solid grey;: Esta propiedad agrega un borde de 0.2% del tamaño de la ventana gráfica (en unidades vh) alrededor del elemento. El color del borde se establece como gris ("grey"). El estilo del borde es "solid", lo que significa que es una línea continua.
* color: white;: Establece el color del texto dentro del elemento como blanco ("white").
* overflow-y: hidden;: Esta propiedad evita que el contenido del elemento se desplace verticalmente si supera la altura del contenedor.



**Selector .HeaderMensInf:**

* height: Establece la altura del elemento en 30 píxeles.
* width: Establece el ancho del elemento al 100% del ancho de su contenedor padre.
* color: Establece el color del texto del elemento en blanco.
* background: Define un degradado de fondo lineal que va desde el color #181818 (negro oscuro) hasta el color #373737 (negro grisáceo) en un ángulo de 180 grados.
* text-align: Alinea el texto del elemento a la derecha.



**Selector .scoreboard:**

* display: flex;: Esta propiedad cambia el modo de visualización del elemento a flex.
* flex-direction: column;: Esta propiedad define la dirección en la que se organizarán los elementos secundarios del elemento .scoreboard.
* align-items: center;: Esta propiedad alinea los elementos secundarios del elemento .scoreboard en el centro del eje horizontal (eje x).

Texto

Descripción generada automáticamente

**Selector .scores:**

* display: flex;: Esta propiedad cambia el modo de visualización del elemento a "flexbox". Flexbox es un modelo de diseño de CSS que permite distribuir los elementos de forma flexible en un contenedor.
* justify-content: space-around;: Esta propiedad define cómo se distribuye el espacio libre entre los elementos flex.
* width: 100%;: Esta propiedad establece el ancho del elemento al 100% del ancho de su contenedor padre.

Texto

Descripción generada automáticamente

**Selector .scores table:**

* border-collapse: collapse: Esta propiedad elimina el borde entre las celdas de la tabla, lo que hace que se vean como una sola unidad.
* width: 45%; Esta propiedad establece el ancho de la tabla en un 45% del ancho de su contenedor principal.
* margin: 10px: Esta propiedad agrega un margen de 10 píxeles alrededor de la tabla.

Texto

Descripción generada automáticamente

**Selector .scores th, .scores td:**

* border: Establece un borde sólido de 1 píxel de color #dcdcdc alrededor de cada celda.
* padding: Agrega un espacio de relleno de 8 píxeles a cada celda, tanto en el eje horizontal como en el vertical.
* text-align: Alinea el texto dentro de cada celda al centro.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

**Selector .scores th:**

* background-color: Esta propiedad define el color de fondo de las celdas de encabezado.
* color: Esta propiedad define el color del texto dentro de las celdas de encabezado.

**DOCUMENTACIÓN CÓDIGO JAVASCRIPT**



* La función utiliza document.getElementById(id) para obtener una referencia al elemento que coincide con el identificador (id) proporcionado.
* Si el id del botón es "BReset" o "BEnter":
  1. Se cambia el color del texto del botón a (#b01010).
  2. Se eliminan los bordes superior e inferior del botón.
  3. Se agrega un temporizador (setTimeout) que después de 150 milisegundos:
     + Cambia el color del texto del botón a un gris oscuro (#390606).
     + Agrega un borde superior fino de color gris oscuro (#171717).
     + Agrega un borde inferior fino de color gris medio (#313131).
* Si el id del botón NO es "BReset" ni "BEnter":
  1. Se cambia el color del texto del botón a (#b01010).
  2. Se agrega un borde superior e inferior de 1 unidad de alto (1vh) y transparentes.
  3. Se agrega un temporizador (setTimeout) que después de 150 milisegundos:
     + Cambia el color del texto del botón a (#390606).
     + Agrega un borde superior fino de color (#171717).
     + Agrega un borde inferior fino de color (#313131).
* Si el id del botón es "BEnter":
  + 1. Se llama a una función ConstructGame() (no documentada aquí). Se presume que esta función inicializa el juego.
* Si el id del botón es "BReset":
  + 1. Se llama a una función Reset() (no documentada aquí). Se presume que esta función reinicia el juego.
* Si el id del botón NO es "BReset" ni "BEnter":
  + 1. Se llama a una función PressKey(button.id) (no documentada aquí). Se presume que esta función maneja el comportamiento específico de cada botón que no sea "BReset" ni "BEnter".



**Declaración de variables:**

* let timeS: Variable para almacenar los segundos del temporizador.
* let timeM: Variable para almacenar los minutos del temporizador.
* let timeH: Variable para almacenar las horas del temporizador.
* let movimientos: Variable global para contar los movimientos del jugador.
* let playerName: Variable global para almacenar el nombre del jugador.

**Evento keydown:**

* Se utiliza la función addEventListener para escuchar eventos de tipo keydown en el documento.
* La función logKey se ejecuta cada vez que se presiona una tecla.

**Función logKey:**

* Esta función recibe el evento de teclado como parámetro (e).
* Dentro de la función, se utiliza un bloque switch para evaluar el código de la tecla presionada (e.code).
* Los casos posibles son:
  + "ArrowUp": **[Flecha arriba]**
  + "ArrowDown": **[Flecha abajo]**
  + "ArrowLeft": **[Flecha izquierda]**
  + "ArrowRight": **[Flecha derecha]**
  + "Enter": **[Enter]**
  + "Escape": **[Escape]**
* Para cada caso, se utiliza la función e.preventDefault() para evitar el comportamiento predeterminado de la tecla (por ejemplo, desplazamiento de la página).



**Declaración de la variable:**

* let button; declara una variable llamada button sin asignarle ningún valor inicial.

**Evento switch:**

* El código utiliza una estructura switch para evaluar el código de la tecla presionada (e.code).
* Cada caso (case) dentro del switch representa un código de tecla específico:
  + ArrowUp: Detecta la tecla "flecha arriba".
  + ArrowDown: Detecta la tecla "flecha abajo".
  + ArrowLeft: Detecta la tecla "flecha izquierda".
  + ArrowRight: Detecta la tecla "flecha derecha".
  + Enter o NumpadEnter: Detecta la tecla "Enter" o "Enter del teclado numérico".
  + Escape: Detecta la tecla "Escape".

**Asignación y llamada a función:**

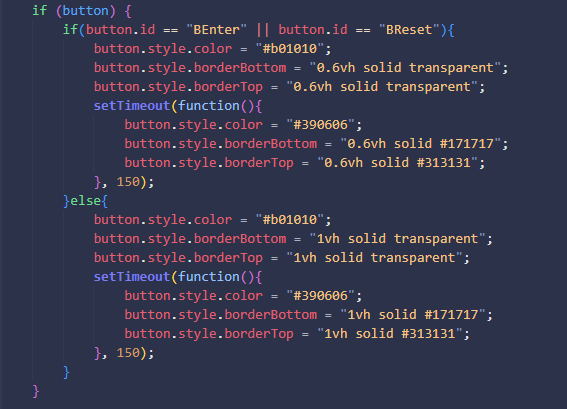
* Dentro de cada caso del switch:
  + Se asigna el elemento del botón correspondiente a la tecla presionada a la variable button. Esto se logra utilizando document.getElementById("BUp") (por ejemplo) para obtener el elemento con el identificador ("id") específico.
  + Se llama a la función PressKey y se le pasa como argumento el id del botón obtenido (button.id).

**Comprobación adicional en "Enter":**

* En el caso de "Enter" o "NumpadEnter", se añade una comprobación adicional con la variable posMenuOP. Si posMenuOP es igual a 0, se llama a la función ConstructGame.

**Caso "Escape":**

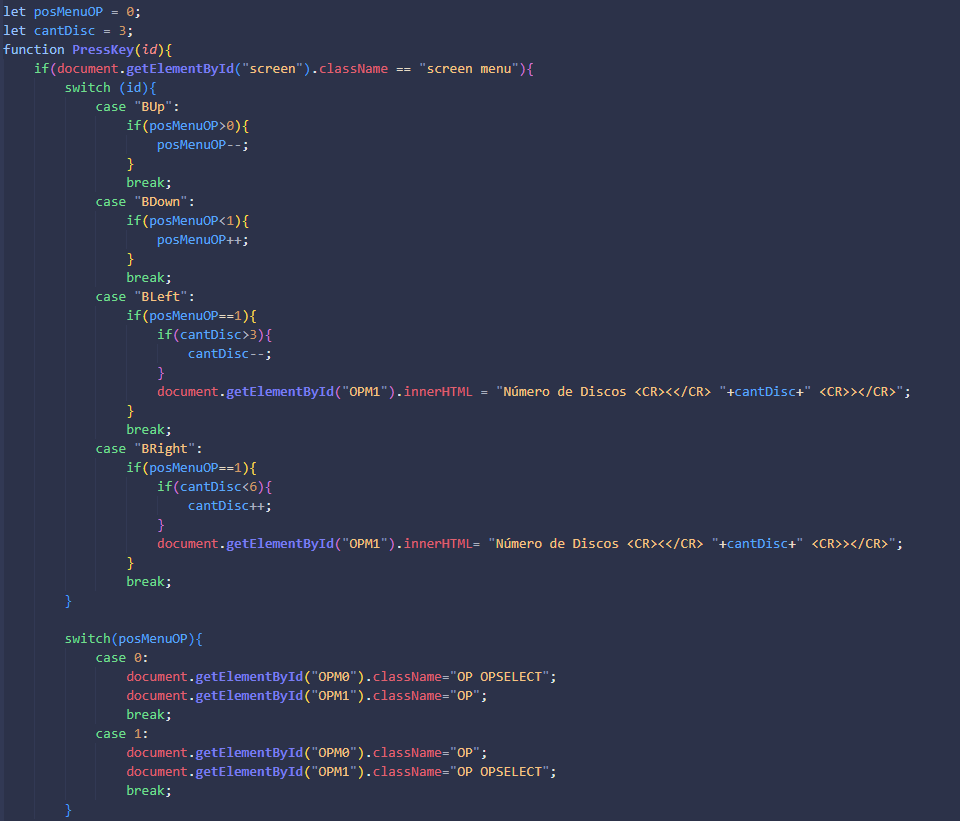
* En el caso de "Escape", se asigna el elemento del botón con identificador "BReset" a la variable button y se llama a la función Reset.



**Verifica el botón:** Primero, se verifica si el elemento button existe.

**Comprueba el identificador:** Si el botón existe, se verifica su identificador (id).

* Si el id es igual a "BEnter" o "BReset":
  + Se cambia el color del texto del botón a (#b01010).
  + Se eliminan los bordes superior e inferior del botón.
  + Se ejecuta una función a los 150 milisegundos que:
    - Cambia el color del texto del botón a (#390606).
    - Se agregan bordes superior e inferior de grosor 0.6vh en color (#171717).
* Si el id es diferente a "BEnter" y "BReset":
  + Se cambia el color del texto del botón a (#b01010).
  + Se eliminan los bordes superior e inferior del botón.
  + Se ejecuta una función a los 150 milisegundos que:
    - Cambia el color del texto del botón a (#390606).
    - Se agregan bordes superior e inferior de grosor 1vh en color (#171717).



**Variables:**

* posMenuOP: Esta variable entera (int) almacena la posición actual en el menú. 0 representa la opción superior y 1 representa la opción inferior.
* cantDisc: Esta variable entera (int) almacena la cantidad de discos configurada actualmente.

**Función PressKey(id):**

* Esta función simula la pulsación de una tecla con el identificador id. Se utiliza para navegar y modificar opciones dentro de un menú en pantalla.

**Comportamiento de la función PressKey(id):**

1. **Verifica el estado de la pantalla:** Primero, la función verifica si el elemento con ID "screen" tiene la clase "screen menu".
2. **Evalúa la tecla presionada:** Si el menú está activo, la función usa un switch para evaluar el valor del parámetro id.
   * BUp (Tecla arriba): Si la posición actual del menú (posMenuOP) es mayor a 0, la posición se decrementa en 1 para navegar hacia la opción superior.
   * BDown (Tecla abajo): Si la posición actual del menú (posMenuOP) es menor a 1, la posición se incrementa en 1 para navegar hacia la opción inferior. (En este caso, solo hay dos opciones en el menú, por lo que este cambio solo ocurre una vez).
   * BLeft (Tecla izquierda): Solo se ejecuta si la posición actual del menú (posMenuOP) es 1 (opción inferior). Si la cantidad de discos (cantDisc) es mayor a 3, se reduce en 1 y se actualiza el contenido del elemento con ID "OPM1" para mostrar la nueva cantidad.
   * BRight (Tecla derecha): Solo se ejecuta si la posición actual del menú (posMenuOP) es 1 (opción inferior). Si la cantidad de discos (cantDisc) es menor a 6, se incrementa en 1 y se actualiza el contenido del elemento con ID "OPM1" para mostrar la nueva cantidad.
3. **Actualiza las opciones del menú:** Luego de evaluar la tecla presionada, la función usa otro switch basado en el valor de posMenuOP para actualizar las clases CSS de los elementos del menú:
   * Si posMenuOP es 0 (opción superior), la clase del elemento con ID "OPM0" se establece en "OP OPSELECT", lo que resalte la opción seleccionada visualmente, y la clase del elemento con ID "OPM1" se establece en "OP", que es el estilo predeterminado para las opciones del menú.
   * Si posMenuOP es 1 (opción inferior), la clase del elemento con ID "OPM0" se establece en "OP", y la clase del elemento con ID "OPM1" se establece en "OP OPSELECT", resaltando la opción inferior seleccionada.



**Comprobación de la pantalla:**

* Se verifica si el elemento HTML con id "screen" tiene la clase "screen".
  + Si es así, el código dentro de este bloque se ejecutará.

**Comprobación del cronómetro:**

* Se verifica si la variable cronometroOn es falsa (!cronometroOn).
  + Si es falsa, significa que el cronómetro no está activo.
  + Se calcula el tiempo inicial (startTime) restando el valor de tiempoTot a la hora actual obtenida con Date.now().
  + Se cambia el valor de cronometroOn a verdadera (true), indicando que el cronómetro está activo.

**Sección switch:**

* Se utiliza una sentencia switch para evaluar el valor de la variable id.
  + Dependiendo del valor de id, se ejecuta un bloque de código específico.

**Casos del switch:**

* BUp:
  + Si el valor de take es igual a "<i class='fa-solid fa-angles-down'></i>" (icono de flecha hacia abajo), se realizan las siguientes acciones:
    - Se imprime el mensaje "takedisc" en la consola.
    - Se llama a la función takeDisc.
    - Se incrementa el valor de la variable movimientos.
* BDown:
  + Si el valor de take es diferente a "<i class='fa-solid fa-angles-down'></i>" (icono de flecha hacia abajo), se realizan las siguientes acciones:
    - Se imprime el mensaje "takedisc" en la consola.
    - Se llama a la función dropDisc.
* BLeft:
  + Si el valor de pointer es mayor a 1, se realizan las siguientes acciones:
    - Se decrementa el valor de pointer en 1.
    - Se llama a la función changePointer para actualizar la posición de un puntero.
* BRight:
  + Si el valor de pointer es menor a 3, se realizan las siguientes acciones:
    - Se incrementa el valor de pointer en 1.
    - Se llama a la función changePointer para actualizar la posición de un puntero.
* BReset:
  + Se llama a la función Reset.

Texto

Descripción generada automáticamente

weightAr: Se inicializa con el valor -1. Representa un peso por defecto o un valor inicial.

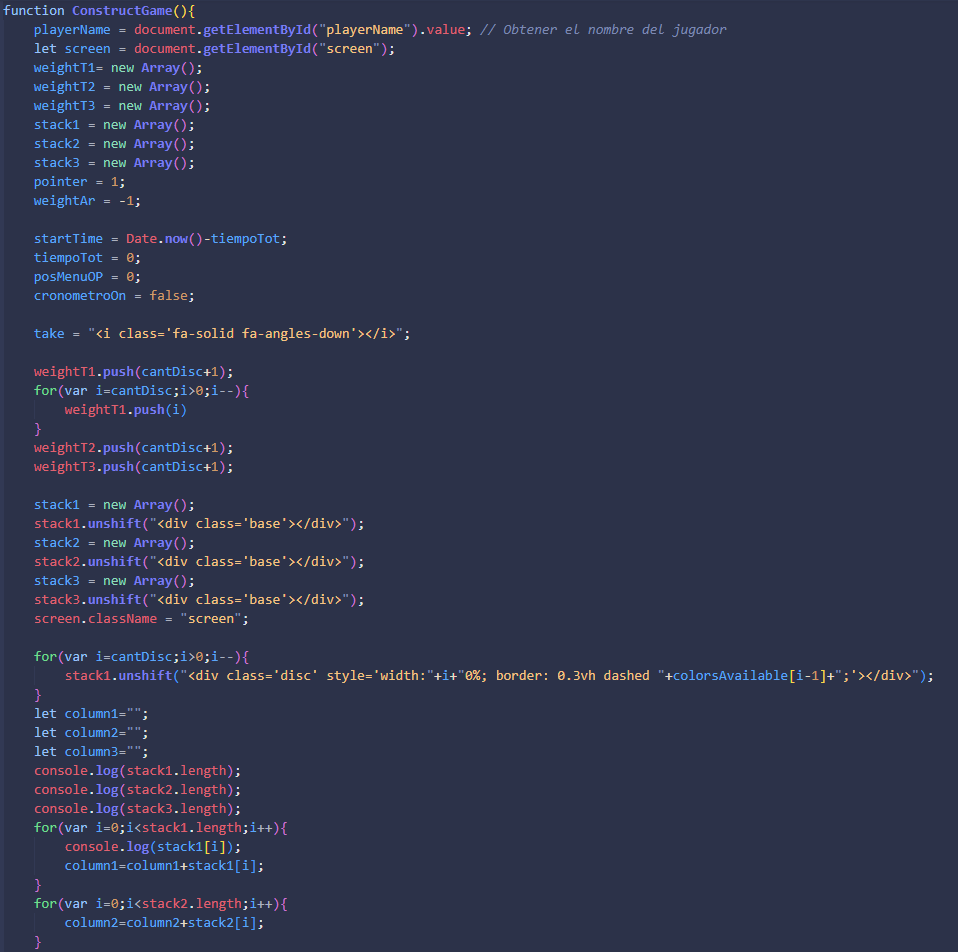
weightT1, weightT2, weightT3: Son arreglos vacíos creados con new Array().

pointer: Se inicializa con el valor 1.

stack1, stack2, stack3: Son también arreglos vacíos creados con new Array().

take: Contiene el código HTML para un ícono de flecha hacia abajo (<i> con clase fa-solid fa-angles-down de Font Awesome).

colorsAvailable: Es un arreglo que contiene una lista de colores en formato hexadecimal.



Texto

Descripción generada automáticamente

Esta función se encarga de inicializar el juego de las Torres de Hanoi. A continuación, se detalla su funcionalidad paso a paso:

**Obtención de datos:**

* Recupera el nombre del jugador del elemento HTML con id "playerName".
* Obtiene una referencia al elemento HTML con id "screen" que representará la pantalla del juego.

**Declaración de variables:**

* weightT1, weightT2, weightT3: Son arrays que almacenarán los pesos de los discos en cada torre (T1: Torre 1, T2: Torre 2, T3: Torre 3).
* stack1, stack2, stack3: Son arrays que contendrán el código HTML para representar las torres con sus respectivos discos.
* pointer: Indica la torre seleccionada actualmente (1 para Torre 1, 2 para Torre 2 y 3 para Torre 3).
* weightAr: Variable auxiliar para el peso del disco a mover (inicializada a -1).
* startTime: Almacena el tiempo inicial del juego utilizando la función Date.now() menos el tiempo total acumulado (tiempoTot).
* tiempoTot: Variable para almacenar el tiempo total transcurrido en el juego (inicializada a 0).
* posMenuOP: Variable posiblemente relacionada con el menú de opciones (valor inicial 0).
* cronometroOn: Variable booleana que indica si el cronómetro está activo (inicializada a falso).
* take: Almacena el código HTML para representar la flecha que indica la torre seleccionada (utiliza el ícono de FontAwesome <i class='fa-solid fa-angles-down'></i>).

**Inicialización de las torres:**

* Se agrega al principio (push) del array weightT1 el valor cantDisc+1 (la cantidad total de discos más 1).
* Se recorre un ciclo desde cantDisc hasta 1, agregando cada valor decreciente a weightT1 (representando el peso de cada disco en la Torre 1).
* Se inicializan weightT2 y weightT3 agregando el valor cantDisc+1 (la cantidad total de discos más 1) a cada array, representando las otras dos torres vacías al inicio.

**Creación de la base de las torres:**

* Se inicializan los arrays stack1, stack2, y stack3 como arrays vacíos.
* Se agrega al principio (unshift) de cada array el código HTML para representar la base de cada torre usando la clase base.

**Creación de los discos en la Torre 1:**

* Se recorre un ciclo desde cantDisc hasta 1, agregando al principio (unshift) de stack1 el código HTML para representar cada disco con la clase disc.
  + El ancho del disco se define con estilo en línea (style='width: + i + '0%;'), basado en el valor de i.
  + El borde de cada disco se define con estilo en línea (border: 0.3vh dashed + colorsAvailable[i-1] + ';), utilizando un borde discontinuo de 0.3vh y un color obtenido de un arraycolorsAvailable`.

**Construcción del HTML para la pantalla del juego:**

* Se crean variables vacías column1, column2, y column3 para almacenar el código HTML de cada torre.
* Se recorre el array stack1 para agregar el código HTML de cada disco de la Torre 1 a la variable column1.
* Se repite el proceso anterior para las torres 2 y 3 con sus respectivos arrays.

**7. Creación del indicador de selección:**

* Se declara una variable arrowPointer para almacenar el código HTML del indicador de selección.
* Se utiliza un switch basado en el valor de pointer para construir el código HTML del indicador:
  + Si pointer es 1, se muestra la flecha apuntando a la Torre 1.
  + Si pointer es 2, se muestra la flecha apuntando a la Torre 2.
  + Si pointer es 3, se muestra la flecha apuntando a la Torre 3.

**8. Creación del HTML completo del juego:**

* Se crea una variable stringHTML para almacenar el código HTML completo de la pantalla del juego.
* Se agrega el código HTML para el temporizador con la clase timer (para muestrar el tiempo transcurrido en el juego).
* Se agrega el código HTML para el contenedor.



**Función changePointer:**

* Si pointer es 1, el HTML resultante tendrá tres elementos div con la clase "arrow". El primer elemento tendrá el contenido definido por take, indicando la posición 1 como la seleccionada.
* Si pointer es 2, el HTML resultante tendrá el elemento div con el contenido take en la posición 2.
* Si pointer es 3, el HTML resultante tendrá el elemento div con el contenido take en la posición 3.



**Variables utilizadas:**

* pointer (número): Indica la pila actual de la que se tomará el disco (1, 2 o 3).
* stack1, stack2, stack3 (arreglos): Representan las pilas de discos 1, 2 y 3 respectivamente.
* weightT1, weightT2, weightT3 (arreglos): Contienen los pesos de los discos en las pilas 1, 2 y 3 respectivamente.
* weightAr (número): Almacena el peso del disco que se toma.

**Comportamiento de la función:**

1. La función verifica si el valor de take es un ícono específico usando Font Awesome (<i> class='fa-solid fa-angles-down'></i>).
2. Se utiliza un switch basado en el valor de pointer para determinar la pila de la que se tomará el disco:
   * case 1: Si pointer es 1, se verifica si stack1 tiene más de un elemento. Si es así, se remueve el primer elemento (shift) y se asigna a take. Además, se llama a la función changePointer y se extrae el último peso de weightT1 y se asigna a weightAr.
   * case 2 y case 3: Similar al caso 1, pero se verifica y opera sobre stack2 y weightT2 para el caso 2, y stack3 y weightT3 para el caso 3.
3. Se imprime información a la consola sobre el peso del disco tomado (weightAr), y los últimos pesos de cada pila (weightT1, weightT2, weightT3).
4. Se llama a la función rewrite para actualizar la interfaz gráfica.



**Función dropDisc()**

**Comportamiento:**

1. **Comprobación inicial:**
   * La función verifica si take no es igual a un código HTML que representa una flecha hacia abajo (<i class='fa-solid fa-angles-down'></i>). Esto se usa para evitar que se mueva un disco vacío.
2. **Movimiento del disco:**
   * Si la condición anterior se cumple, la función procede a mover el disco según el valor de pointer:
     + case 1 (torre 1 como origen):
       - Si el peso del disco superior en la torre 1 (weightT1[weightT1.length-1]) es mayor que el peso del disco a mover (weightAr), se realizan las siguientes acciones:
         * Se agrega el peso del disco a mover al array weightT1.
         * Se establece weightAr a -1 (para indicar que el disco ya se movió).
         * Se agrega el valor de take (que representa el disco) al comienzo del array stack1 (para mantener un registro del historial de movimientos).
         * Se establece take a un valor que indica un disco vacío.
         * Se llama a la función changePointer() para actualizar la torre destino.
     + Los casos 2 y 3 siguen una lógica similar, verificando los pesos de los discos superiores en las torres 2 y 3 respectivamente, y realizando movimientos similares si las condiciones se cumplen.
3. **Impresión de información:**
   * Se imprime información al console.log:
     + El peso del disco a mover (weightAr).
     + Los pesos de los discos superiores en cada torre (weightT1[weightT1.length-1], etc.).
4. **Actualización de la visualización:**
   * Se llama a la función rewrite() para actualizar la visualización del juego en la interfaz de usuario.
5. **Comprobación de fin del juego:**
   * La función verifica si la longitud del array weightT3 (que representa la torre destino final) es igual a cantDisc + 1. Esto indica que se han movido todos los discos a la torre final. Si se cumple la condición, se llama a la función ENDGAME() para finalizar el juego.



**Función rewrite:**

**Comportamiento:**

1. **Inicialización de variables:**
   * Se crean tres cadenas de texto vacías: column1, column2 y column3. Estas variables se usarán para almacenar el contenido que se va a mostrar en cada columna.
2. **Recorrido de las listas:**
   * La función utiliza tres bucles for para iterar sobre las listas stack1, stack2 y stack3.
   * Dentro de cada bucle, se concatena (une) el contenido de cada elemento de la lista a la variable column correspondiente. Por ejemplo, en el primer bucle, se va agregando el contenido de cada elemento de stack1 a la variable column1.
3. **Actualización del elemento HTML:**
   * Se obtiene una referencia al elemento HTML con el identificador "screenGame" utilizando el método document.getElementById().
   * Se establece la propiedad innerHTML del elemento con una cadena de texto que contiene el código HTML para crear tres elementos <div>.
   * Cada elemento <div> tiene un identificador único (post1, post2 o post3) y una clase post.
   * El contenido de cada <div> se rellena con el contenido correspondiente de las variables column1, column2 y column3.



**Función ENDGAME:**

**Comportamiento:**

1. La función desactiva el cronómetro (cronometroOn = false).

2. Obtiene una referencia al elemento HTML con identificador "screen" y lo almacena en la variable screen.

3. Cambia la clase CSS del elemento screen a "screen endGame". Esto cambia la apariencia del elemento para mostrar la pantalla de fin de juego.

4. Construye el contenido HTML que se mostrará en la pantalla de fin de juego. Este contenido incluye:

* Un título "END GAME"
* El tiempo total del juego en verde, formateado con la función formatearMS.
* El número total de movimientos realizados en verde.
* Un mensaje que indica que se puede volver al menú presionando Escape.

5. Se agrega el contenido HTML construido con anterioridad al elemento screen usando la propiedad innerHTML.

6. Obtiene referencias a los elementos HTML con identificadores "timeTable" y "movesTable". Luego, accede a sus elementos "tbody" (cuerpo de la tabla).

7. Inserta una nueva fila en cada cuerpo de tabla (timeTableBody y movesTableBody).

8. Inserta dos celdas en cada fila nueva. La primera celda contendrá el nombre del jugador y la segunda el tiempo o los movimientos totales, respectivamente.

9. Se asigna el nombre del jugador a las celdas correspondientes a los nombres.

10. Se asigna el tiempo total formateado (con la función formatearMS) y el número de movimientos a las celdas correspondientes a los valores.



**Función Reset:**

**Comportamiento:**

1. **Obtener elemento "screen"**: Se obtiene una referencia al elemento HTML con identificador "screen" y se almacena en la variable screen.

2. **Cambiar clase del elemento "screen"**: Se modifica la clase del elemento "screen" a "screen menu".

3. **Actualizar contenido del elemento "screen"**: Se modifica el contenido HTML del elemento "screen" creando una estructura con varios elementos:

* Un botón con clase EnterResetButtons y estilo de color verde (color:#39ca00). El botón tiene un identificador "infoL" y al hacer clic ejecuta la función MessageInfo().
* Un texto que indica "Instrucciones del juego".
* Un elemento con clase OP OPSELECT e identificador OPM0 que contiene el texto "Iniciar".
* Un elemento con clase OP e identificador OPM1 que contiene el texto "Número de Discos" seguido de tres símbolos "<" y ">" y el valor actual (3).
* Una etiqueta para un campo de texto con el nombre "Jugador" y el identificador "playerName".
* Un campo de texto para ingresar el nombre del jugador con identificador "playerName" y el atributo required indicando que es obligatorio llenarlo.

Texto

Descripción generada automáticamente

**Función formatearMS:**

**Comportamiento:**

1. **Obtener los milisegundos:**

* Se obtiene el resto de la división de tiempo\_ms entre 1000 y se almacena en la variable MS. Este valor representa los milisegundos restantes después de extraer las unidades de segundos completos.

2. **Calcular unidades de tiempo:**

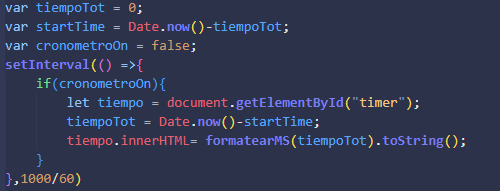
* Se calcula el número total de segundos dividiendo tiempo\_ms entre 1000 y se almacena en la variable total\_seconds.
* Se calcula el número total de minutos dividiendo total\_seconds entre 60 y se almacena en la variable total\_minutes.
* Se calcula el número total de horas dividiendo total\_minutes entre 60 y se almacena en la variable total\_horas.

3. **Formatear la salida:**

* Se obtiene el resto de la división de total\_seconds entre 60 y se convierte a cadena de texto. Se almacena en la variable display\_seconds.
* Se convierte total\_minutes a cadena de texto y se almacena en display\_minutes.
* Se convierte total\_horas a cadena de texto y se almacena en display\_horas.
* Se utilizan métodos de relleno (padStart) para garantizar que display\_seconds, display\_minutes y display\_horas tengan siempre una longitud de dos caracteres rellenando con ceros a la izquierda si es necesario.

4. **Retorno:**

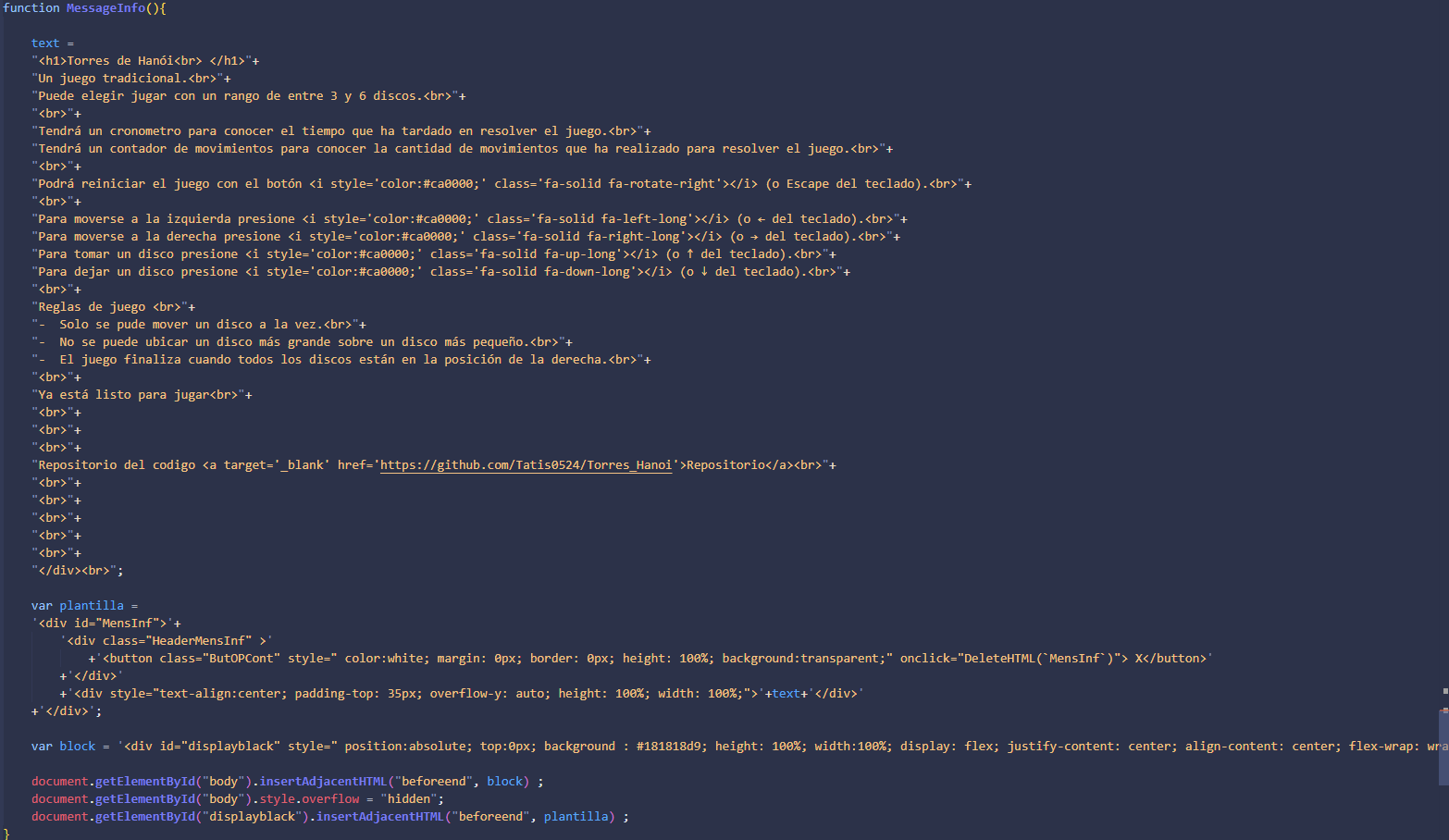
* La función retorna una cadena formateada que combina las variables display\_horas, display\_minutes, display\_seconds y MS separadas por dos puntos (:) y un punto (.) para representar el formato "hh:mm:ss.ms".



1. **Variables:**

* tiempoTot: Esta variable almacena el tiempo total transcurrido en milisegundos.
* startTime: Esta variable almacena la marca de tiempo en milisegundos cuando se inicia el cronómetro.
* cronometroOn: Esta variable indica si el cronómetro está activo o no.

1. **Función setInterval**: Esta función se utiliza para ejecutar una función repetidamente a un intervalo especificado.
2. **Función anónima**: La función anónima dentro del setInterval verifica si el cronometroOn está configurado en true. Si es así, actualiza el texto del elemento HTML con ID "timer" con el tiempo transcurrido formateado por la función formatearMS.



**Función** **MessageInfo**:

**Comportamiento**:

1. **Creación del texto del mensaje:**

* Se define una variable text que contiene el HTML del mensaje que se mostrará en el modal.
* El mensaje incluye:
  + El título del juego "Torres de Hanói".
  + Una breve descripción del juego.
  + Instrucciones sobre cómo jugar, incluyendo los controles del teclado y las reglas del juego.
  + Un enlace al repositorio del código del juego en GitHub.

2. **Creación de la plantilla HTML del modal:**

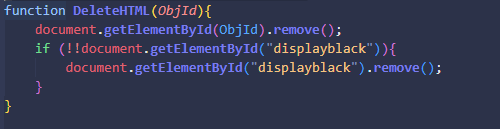
* Se define una variable plantilla que contiene el HTML de la estructura del modal.
* La plantilla incluye:
  + Un contenedor principal con el ID "MensInf".
  + Un encabezado con un botón para cerrar el modal (X).
  + Un contenedor para el contenido del mensaje (con overflow-y: auto para permitir el desplazamiento si es necesario).

3. **Creación del bloque negro que oscurece la pantalla:**

* Se define una variable block que contiene el HTML de un bloque negro que se superpone a toda la pantalla.
* El bloque negro tiene un estilo que lo hace semitransparente y lo centra en la pantalla.

4. **Inserción del HTML en el DOM:**

* Se utiliza el método insertAdjacentHTML para insertar el HTML del bloque negro y la plantilla del modal en el elemento con ID "body".
* Se establece el estilo overflow: hidden en el elemento "body" para evitar que el contenido de la página se desplace detrás del modal.



**Función DeleteHTML:**

**Comportamiento:**

1. La función DeleteHTML recibe como parámetro un ID de elemento HTML.

2. La función utiliza el método getElementById para obtener una referencia al elemento HTML con el ID especificado.

3. Si el elemento se encuentra, el método remove() se utiliza para eliminarlo del DOM.

4. La función comprueba si existe un elemento con ID "displayblack".

5. Si el elemento "displayblack" existe, se elimina utilizando el método remove().