El futuro digital es de todos

MinTIC

Ciclo 3:

Desarrollo de Software

UNIVERSIDAD NACIONAL DE COLOMBIA

MisionTIC2022

VERSIÓN 1.0

Unidad de educación continua y permanente
Facultad de Ingeniería

Unidad Camilo Torres
Calle 44 # 45-67
Bloque B5 piso 1

(57) 1 316 5000
uec_fibog@unaleduco

Componente de Presentación

(JavaScript)

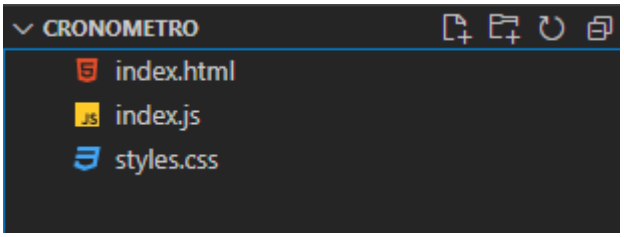
Actividad Práctica

Para llevar a cabo el desarrollo del componente correspondiente a la capa de presentación es necesario conocer algunas tecnologías como *HTML*, *CSS* y *JavaScript*, las dos primeras fueron vistas en la guía anterior, en esta guía se realizará un ejercicio práctico cuyo objetivo es mostrar la manera en la que *JavaScript* dota de *interactividad* a *HTML* y *CSS*.

El ejercicio consistirá en crear un *cronometro* totalmente funcional el cual tendrá tres botones (*play*, *pause* y *reset*), para lograr este objetivo será necesario el uso de *funciones*, *condicionales*, *variables*, *objetos*, *manejo de eventos* y lo más importante el acceso al *DOM* a través del objeto document.

Desarrollo del Cronómetro

Antes de iniciar el desarrollo del cronómetro se debe crear la siguiente *estructura* de *archivos*:



Dado que ya se tiene un dominio básico de HTML y CSS, no se explicará en detalle el contenido de los archivos *index.html* y *styles.css*.

HTML

El contenido del archivo *index.html* es el siguiente:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="stylesheet" href="styles.css">

    <title>Mi Cronometro</title>

</head>
<body>
    <div class="container">
```

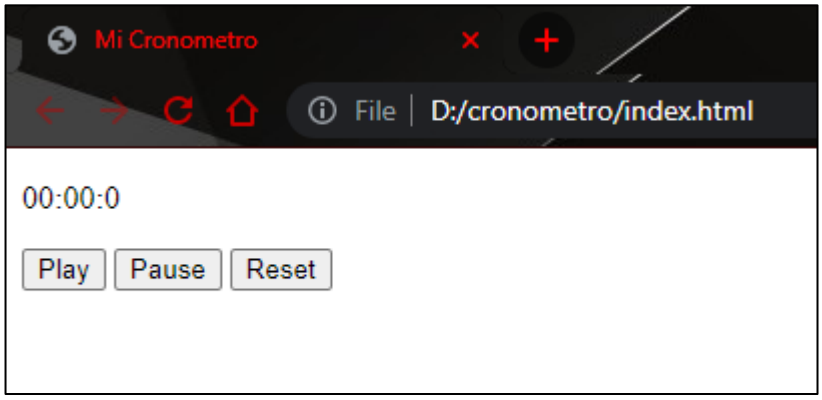
```
        <div class="cronometro">
            <div class="tablero">
                <p id="tablero">00:00:0</p>
            </div>
            <div class="botones">
                <button id="boton_play">Play</button>
                <button id="boton_pause">Pause</button>
                <button id="boton_reset">Reset</button>
            </div>
        </div>
    </div>

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

En este archivo se define el esqueleto del cronómetro, si bien es bastante sencillo se deben matizar algunos detalles:

- La línea <link rel="stylesheet" href="styles.css"> permite agregar al documento *HTML* el archivo *CSS* que contendrá los estilos correspondientes.
- La línea <script src="index.js"></script> permite agregar al documento *HTML* el archivo *JavaScript*. Exeisten muchas maneras de utilizar *JavaScript* dentro de un documento *HTML*, pero esta es la mejor manera, ya que permite separar los códigos de manera clara y ordenada en *archivos distintos*. Por otro lado, se debe notar que la ubicación de la línea es casi al final del documento, esto se debe a que es una buena práctica importar o asociar el script cuando ya se ha definido todo el documento *HTML*.
- Si bien se tiene una gran cantidad de *divs*, el objetivo de estos es meramente estético, no tienen ninguna relevancia a la hora de trabajar con JavaScript, los elementos importantes son el texto del *tablero* y los *tres botones*.
- Se debe notar que en algunos casos se hace uso de *clases* y en otros de *ids*, las clases serán utilizadas en la definición de estilos, mientras los *ids* serán utilizados en *JavaScript*.

Para visualizar el avance se debe abrir el archivo *index.html* desde un *navegador*, y de momento el cronómetro lucirá así:



CSS

El contenido del archivo *styles.css* es el siguiente:

```css
*{
    box-sizing: border-box;
    margin: 0;
    padding: 0;
}

.container{
    width: 100vw;
    height: 100vh;
    background-color: darkcyan;
    display: flex;
    justify-content: center;
    align-items: center;
}

.cronometro {
    width: 40%;
    height: 60%;
    background-color: white;
    border: 3px solid rgba(0, 0, 0, 0.8);
    display: flex;
    align-items: center;
    justify-content: center;
    flex-direction: column;
}


.tablero{
    width: 60%;
    height: 50%;
    background-color: rgba(0, 0, 0, 0.7);
    border: 3px solid rgba(0, 0, 0, 0.8);
    display: flex;
    align-items: center;
    justify-content: center;
}

.tablero p {
    color: white;
    font-size: 100px;
    display: flex;
    align-items: center;
    justify-content: center;
```

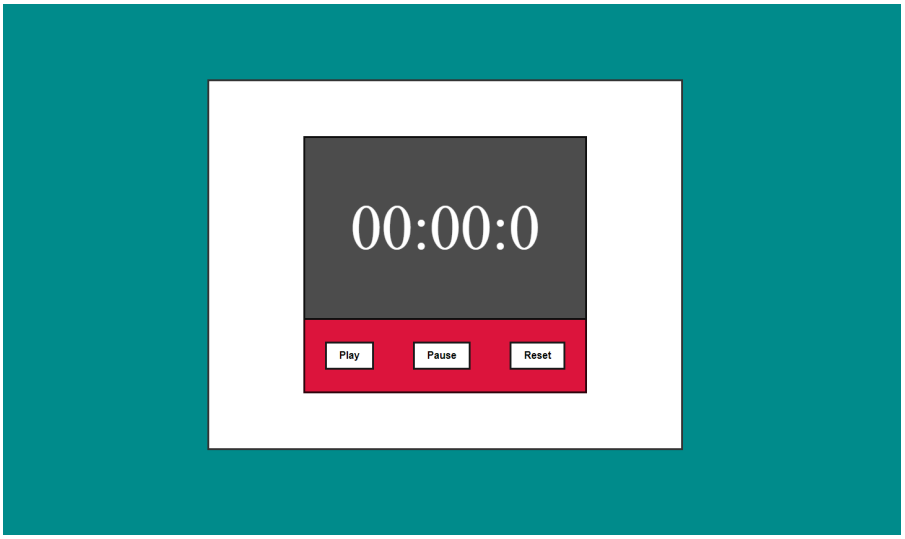
```
}

.botones{
    width: 60%;
    height: 20%;
    background-color: crimson;
    border: 3px solid rgba(0, 0, 0, 0.8);
    border-top: none;
    display: flex;
    align-items: center;
    justify-content: space-around;
}

.botones button{
    background-color: white;
    padding: 10px 20px;
    border: 3px solid rgba(0, 0, 0, 0.9);
    color: black;
    font-weight: bold;
    font-size: 16px;
}

.botones button:hover{
    background-color: white;
    color: red;
}
```

Este archivo simplemente contiene una serie de instrucciones para dotar de *estilo* al *cronómetro*, con los estilos agregados el cronómetro lucirá así:



JavaScript

Ahora que se tiene el esqueleto y los estilos del *cronómetro*, se procederá a implementar sus funcionalidades, estas serán desarrolladas poco a poco y explicando en detalle cada una de las partes del código que deben ir en el archivo *index.js*

DOM: Obtener elementos

La *primera parte* del código del archivo *index.js* es la siguiente:

```
// DOM: Obtener elementos
let tablero     = document.getElementById("tablero");
let boton_play  = document.getElementById("boton_play");
let boton_pause = document.getElementById("boton_pause");
let boton_reset = document.getElementById("boton_reset");
```

Dentro del *HTML* del cronómetro existen cuatro elementos principales, que son el *tablero*, en el cual se muestra el tiempo y los *tres botones*, para poder trabajar con ellos a lo largo del código es necesario obtenerlos desde el *DOM* y guardarlos en una variable, esto se logra con ayuda de un método del objeto *document* y el *identificador* (Id) de cada uno de los elementos a capturar.

Estados Cronómetro

La segunda parte del código del archivo *index.js* es la siguiente:

```
// Estados Cronometro
let esta_activo =  false;
let time = {
    decimas     : 0,
    segundos    : 0,
    minutos     : 0
}
```

Para poder controlar el funcionamiento del cronometro es necesario definir dos estados, el primero es una *variable booleana* llamada esta_activo que permitirá determinar si el cronómetro está pausado o está activo, el segundo estado es un *objeto* que almacenará el *tiempo* transcurrido, si bien podrían haberse guardado el tiempo en variables independientes, la opción de usar un objeto permite mantener la información ordenada, el estado inicial del cronómetro será estar detenido y el tiempo en cero.

Función Actualizar

La tercera parte del código del archivo *index.js* es la siguiente:

```
// Funcion Actualizar
function formato(numero){
    if(numero<10){
        return "0"+numero;
    }
    else{
        return numero;
    }
}



function actualizar(){

    time.decimas++;

    if(time.decimas == 10){
        time.decimas = 0;
        time.segundos++;
    }

    if(time.segundos == 60){
        time.segundos = 0;
        time.minutos++;
    }

    tablero.innerHTML = `${formato(time.minutos)}:${formato(time.segundos)}:${time.decimas}
`


    if(esta_activo == true){
        setTimeout(actualizar,100);
    }
}
```

En este fragmento de código se define la parte principal del cronómetro, primero se tiene una *función auxiliar* llamada *formato*, esta función tiene como objetivo *transformar* dígitos como “8” a “08”, esto resulta útil para darle el formato adecuado al tiempo.

La función actualizar tiene 3 partes:

- Primero, dado que el *cronómetro avanza* en *décimas* (cada 100 milisegundos) lo que se hace es incrementar el valor de las décimas en uno, esto se realiza con ayuda del operador de incremento. Una vez se ha actualizado el valor de las décimas, se comprueba con condicionales que no se hayan pasado los límites.
- Luego de realizar las comprobaciones se procede a modificar el valor del tablero con el nuevo tiempo, esto se logra con la funcionalidad *innerHTML* y el elemento capturado. Puede parecer

un poco complicado construir el nuevo valor, pero realmente lo único que se está haciendo es concatenar los valores del tiempo en un solo *string*.

- Por último se realiza una comprobación para saber si la función actualizar debe ser nuevamente utilizada en caso de que el cronómetro este activo, esto se logra con ayuda de la función predeterminada *setTimeout* con la cual se programa una nueva ejecución dentro de una décima de segundo.

Funciones Botones

La cuarta parte del código del archivo *index.js* es la siguiente:

```
// Funciones Botones
function play(){
    if(esta_activo == false){
        esta_activo = true;
        actualizar();
    }
}

function pause(){
    esta_activo = false;
}


function reset(){
    time.decimas    = 0;
    time.segundos   = 0;
    time.minutos    = 0;

    tablero.innerHTML = `${formato(time.minutos)}:${formato(time.segundos)}:${time.decimas}
`
}
```

Ahora que se tiene la función principal implementada, se procede a implementar las funcionalidades para cada uno de los botones, estos códigos son bastantes simples:

- La *función play* únicamente cambia el estado del cronómetro y ejecuta la función actualizar para que el tiempo comience a avanzar.
- La *función pause* simplemente cambia el estado del cronómetro, para detener las actualizaciones del tiempo.
- La *función reset* actualiza los valores del objeto time a cero y luego actualiza el valor en el tablero.

Escuchar Eventos

La quinta parte del código del archivo *index.js* es la siguiente:

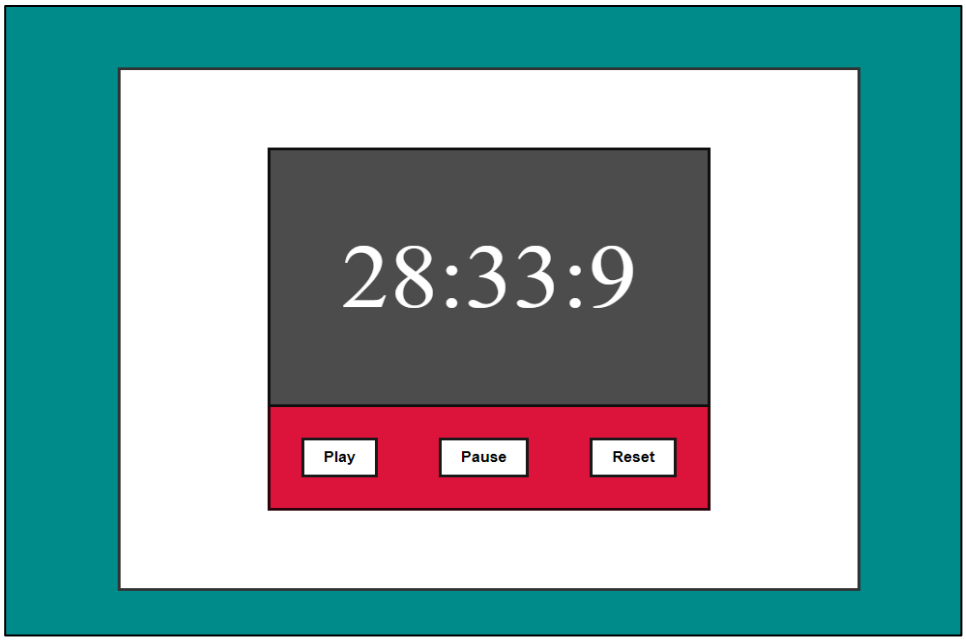
```
// Escuchar Eventos
boton_play.addEventListener('click', play);
boton_pause.addEventListener('click', pause);
boton_reset.addEventListener('click', reset);
```

Por último se tiene que asociar las *funcionalidades* implementadas con sus respectivos *botones* a través de un *evento*, es decir cuando un *botón* produzca un evento de tipo *clic* se hará una llamada a su respectiva función.

Este proceso completa la interacción entre *HTML* y *JavaScript*, primero *JavaScript* captura los *eventos* para cada uno de los botones y le asocia una función que actualiza de manera *directa* o *indirecta* los *estados* del cronómetro, cuando se han actualizado estos estados, *JavaScript* actualiza los correspondientes valores en el *tablero*.

Resultado Final

El resultado final será:



Nota: de ser necesario, en el material de la clase se encuentra una archivo *C3.AP.13. js.rar*, con el desarrollo del ejercicio realizado en esta guía.