21-5-2021

GUÍA IMPLEMENTACIÓN SCRIPT-HID.JS

ALBERTO AYUSO MARTÍN-PORRO

HOMERIA OPEN SOLUTIONS (Cáceres)

ÍNDICE

1. INTRODUCIÓN

Un dispositivo de interfaz humana o HID es un tipo de dispositivo informático que interactúa directamente con humanos, tanto al recibir información de los mismos como proporcionándosela. El término “HID” se refiere más comúnmente a la especificación USB-HID.

HID consta de dos conceptos fundamentales: informes y descriptores de informes. Los *informes* son los datos que se intercambian entre un dispositivo y un cliente de software. El *descriptor de informe* describe el formato y el significado de los datos que admite el dispositivo.

Un HID es un tipo de dispositivo que toma entradas de personas o proporciona salidas a ellas. También se refiere al protocolo HID, un estándar para la comunicación bidireccional entre un host y un dispositivo que está diseñado para simplificar el procedimiento de instalación. El protocolo HID se desarrolló originalmente para dispositivos USB, pero desde entonces se ha implementado en muchos otros protocolos, incluido Bluetooth.

Las aplicaciones y los dispositivos HID intercambian datos binarios a través de tres tipos de informes:

Tipo de informe	Descripción
Informe de entrada	Datos que se envían desde el dispositivo a la aplicación (por ejemplo, se presiona un botón).
Informe de salida	Datos que se envían desde la aplicación al dispositivo (por ejemplo, una solicitud para encender la luz de fondo del teclado).
Informe de funciones	Datos que pueden enviarse en cualquier dirección. El formato es específico del dispositivo.

Un descriptor de informe describe el formato binario de los informes admitidos por el dispositivo. Su estructura es jerárquica y puede agrupar informes como colecciones distintas dentro de la colección de nivel superior. El formato del descriptor está definido por la especificación HID.

Un uso de HID es un valor numérico que se refiere a una entrada o salida estandarizada. Los valores de uso permiten que un dispositivo describa el uso previsto del dispositivo y el

propósito de cada campo en sus informes. Por ejemplo, se define uno para el botón izquierdo de un mouse. Los usos también se organizan en páginas de uso, que proporcionan una indicación de la categoría de alto nivel del dispositivo o informe.

Usando la API de WebHID

La API WebHID se basa en gran medida en JavaScript Promises . Si no está familiarizado con ellos, consulte este excelente tutorial de Promesas . Una cosa más, () => {} son simplemente funciones de flecha de ECMAScript 2015.

2. IMPLEMENTACIÓN DEL FICHERO SCRIPT-HID.JS DE NUESTRO PROYECTO

2.1 Importaciones:

Importamos los colores que vayamos a necesitar del fichero **color.js**:

```
import { GREEN, OFF, RED } from '../colors.js';
```

Con esta línea conseguimos acceder a los colores declarados en el fichero color.js y poder utilizarlos en nuestro script.

2.2 Declaraciones:

Lo primero que hacemos es declarar una constante, la cual, almacena los identificadores de los dispositivos que vayamos a utilizar:

```
const filters = [
  {
    //identificadores de BlinkStick
    vendorId : 0x20A0 ,
    productId : 0x41e5
  },
  {
    // identificadores de Blink(1) mk2
    vendorId : 0x27b8,
    productId : 0x01ed
  }
];
```

A continuación, se muestran varias variables que utilizaremos a modo de simulación, serán sustituidas por funciones de la aplicación a la que queramos integrar los dispositivos.

```
let DEVICE = 0; // 0-> BlinckStick, 1 -> Blink(1) mk2,
 ponemos por defecto BlinkStick.
let LED_COUNT = 8; // Número de led del dispositivo, ponemos por defecto
8 que son los que tiene BlinkStick.
```

```
let simulation = 1; // Utilizaremos esta variable para simular cuando nos
 llegue una notificacion a la pagina de burguer king de pedido correcto o
 pedido cancelado. 0 -> Cancelado.  1 -> Correcto.
let intermittent = 0; // Utilizaremos esta variable para simular efecto i
ntermitente o fijo. 0 -> Fijo.  1 -> Intermitente.

let running = false; // Bandera para controlar cuando queramos apagar el
dispositivo.
let disconnect = false; // Bandera para controlar cuando queramos descone
ctar el dispositivo.
```

DEVICE: Los dispositivos HID que vamos a utilizar, en nuestro caso son dos; BlinckStick y Blink(1) mk2.

LED_COUNT: Número de led que tiene el dispositivo. BlinckStick tiene 8 leds y Blink(1) mk2 tiene 1 led.

Simulation: Esta variable es utilizada para simular cuando llega un pedido correcto o cancelado, es decir, si llega correcto(simulation==1) la iluminación es de color verde y si es un pedido cancelado(simulation==0) la iluminación es de color rojo.

Intermittent: Esta variable es utilizada para simular cuando llega notificación a gusto del cliente, es decir, iluminación intermitente (Intermittent ==1) la iluminación es a ráfagas (el intervalo de ráfagas se puede poner a gusto del cliente, en esta prueba es de 500ms) e iluminación fija (Intermittent ==0) la iluminación se mantiene fija.

Running: Es utilizada para detener la simulación, es decir, running se encuentra a true y cuando el usuario decide parar la simulación, running pasa a false.

Disconnect: Igual que running solo que para desconectar el dispositivo que se encuentra actualmente conectado.

2.3 Implementación de los botones del fichero HTML

Básicamente lo que conseguimos con estas líneas de código es decir que es lo que se debe ejecutar en cada momento cuando el usuario pulsa algún botón:

```
document.querySelector('#ConnectDevice').addEventListener('click', Connec
t);
document.querySelector('#DisconnectDevice').addEventListener('click',  ()
 => (disconnect = true, document.getElementById("nom-
dispositivo").innerHTML =''));

document.querySelector('#pedido_correcto').addEventListener('click', () =
> (simulation = 1));
document.querySelector('#pedido_cancelado').addEventListener('click', ()
=> (simulation = 0));
```

```
document.querySelector('#start-
LuzFija').addEventListener('click',() => (intermittent = 0));
document.querySelector('#start-
LuzIntermitente').addEventListener('click', () => (intermittent = 1));

document.querySelector('#start').addEventListener('click', Start );
document.querySelector('#stop').addEventListener('click', () => (running
= false));
```

2.4 Funciones implementadas:

async function Connect() : Se encarga de conectar el dispositivo seleccionado por el usuario y lo devuelve.

async function Start() : Comienza la simulación con el dispositivo conectado y hasta que el usuario no decida parar la simulación o desconectar el dispositivo la simulación continuará ejecutandose.

async function getOpenedDevice() : Esta función es la encargada de detectar lo dispositivos HID que hay conectados, en caso de no haber ninguno, te muestra un dialogo donde filtra por la constante *filtres* que declaramos al principio del scrip. Por tanto, esta función solo te permite conectar los dispositivos enchufados a tu pc que coincidan con los ids almacenados en la constante filters.

async function setColor(device, index, [r, g, b], retries = 1) : Función encargada de asignar el brillo de la iluminación de los leds y un color específico pasado por parámetros.

function wait(ms) : Devuelve una nueva promesa y espera los ms pasados por parámetros.

async function clear(device) : Limpia el dispositivo pasado por parámetros, es decir, pone color 0 a todos los led (apaga los leds).

function* permanent() : En esta función establecemos las características de la simulación (colores, intervalo de tiempo entre ráfagas, etc). Es llamada en el método **Start().** Para más información solo function* pulse aquí.