Debajo del código, encontrarás la explicación detallada de la lógica paso a paso.

Guía de Estudio: JavaScript Drum Kit

1. El Código Completo (Unificado)

Puedes guardar esto como index.html.

HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>JavaScript Drum Kit</title>
    <link rel="icon" href="https://fav.farm/guitar" />

    <style>
        /* --- INICIO DEL CSS --- */
        html {
                font-size: 10px;
                /* Nota: Asegúrate de tener la imagen o cambiar la URL */
                background: url('./electric-guitar-with-neon-light-still-life.jpg') bottom center;
                background-size: cover;
        }

        body, html {
                margin: 0;
                padding: 0;
                font-family: sans-serif;
        }

        .keys {
                display: flex;
                flex: 1; /* Corregido ligeramente para mejor compatibilidad */
                min-height: 100vh;
                align-items: center;
                justify-content: center;
        }

        .key {
                border: .4rem solid white;
```

```
            border-radius: .5rem;
            margin: .1rem;
            font-size: 1.5rem;
            transition: all .07s ease; /* CLAVE: Esto suaviza el golpe visual */
            width: 10rem;
            text-align: center;
            color: white;
            background: rgb(0,0,0,0.4);
            text-shadow: 0 0 0.8rem darkred;
        }

        /* Esta clase se añade con JS cuando tocas la tecla */
        .playing {
            transform: scale(1.1);
            border-color: red;
            box-shadow: 0 0 1rem red;
        }

        kbd {
            display: block;
            font-size: 4rem;
        }

        .sound {
            font-size: 1.2rem;
            text-transform: uppercase;
            letter-spacing: .1rem;
            color: white;
        }
        /* --- FIN DEL CSS --- */
    </style>
</head>
<body>

    <div class="keys">
        <div data-key="65" class="key">
            <kbd>A</kbd>
            <span class="sound">clap</span>
        </div>
        <div data-key="83" class="key">
            <kbd>S</kbd>
            <span class="sound">hihat</span>
        </div>
        <div data-key="68" class="key">
```

```
            <kbd>D</kbd>
            <span class="sound">kick</span>
        </div>
        <div data-key="70" class="key">
            <kbd>F</kbd>
            <span class="sound">openhat</span>
        </div>
        <div data-key="71" class="key">
            <kbd>G</kbd>
            <span class="sound">boom</span>
        </div>
        <div data-key="72" class="key">
            <kbd>H</kbd>
            <span class="sound">ride</span>
        </div>
        <div data-key="74" class="key">
            <kbd>J</kbd>
            <span class="sound">snare</span>
        </div>
        <div data-key="75" class="key">
            <kbd>K</kbd>
            <span class="sound">tom</span>
        </div>
        <div data-key="76" class="key">
            <kbd>L</kbd>
            <span class="sound">tink</span>
        </div>
    </div>

    <audio data-key="65" src="sounds/clap.wav"></audio>
    <audio data-key="83" src="sounds/hihat.wav"></audio>
    <audio data-key="68" src="sounds/kick.wav"></audio>
    <audio data-key="70" src="sounds/openhat.wav"></audio>
    <audio data-key="71" src="sounds/boom.wav"></audio>
    <audio data-key="72" src="sounds/ride.wav"></audio>
    <audio data-key="74" src="sounds/snare.wav"></audio>
    <audio data-key="75" src="sounds/tom.wav"></audio>
    <audio data-key="76" src="sounds/tink.wav"></audio>

<script>
    /* --- INICIO DE LA LÓGICA JAVASCRIPT --- */

    // Función principal: Tocar sonido y activar animación
    function playSound(element) {
```

```
        // 1. Busca el audio y la tecla visual usando el código de la tecla presionada (keyCode)
        const audio = document.querySelector(`audio[data-key="${element.keyCode}"]`);
        const key = document.querySelector(`.key[data-key="${element.keyCode}"]`);

        // 2. Si no hay audio para esa tecla, detenemos la función
        if (!audio) return;

        // 3. Rebobina el audio al inicio (para poder tocarlo muchas veces seguidas rápidamente)
        audio.currentTime = 0;

        // 4. Reproduce el sonido
        audio.play();

        // 5. Añade la clase CSS para el efecto visual
        key.classList.add('playing');
    }

    // Función secundaria: Limpiar la animación
    function removeTransition(element) {
        // Solo nos importa si lo que terminó de animarse fue el 'transform' (el crecimiento)
        if (element.propertyName !== 'transform') return;

        // 'this' se refiere al elemento (la tecla) que disparó el evento. Le quitamos la clase.
        this.classList.remove('playing');
    }

    // CONFIGURACIÓN INICIAL (Se ejecuta al cargar la página)

    // A. Seleccionamos todas las teclas visuales
    const keys = document.querySelectorAll(`.key`);

    // B. A cada tecla le añadimos un "oído" para saber cuándo termina su transición CSS
    keys.forEach(key => key.addEventListener('transitionend', removeTransition));

    // C. Escuchamos el teclado globalmente para disparar el sonido
    window.addEventListener('keydown', playSound);

</script>

</body>
</html>
```

---

2. Explicación de la Lógica (Cómo funciona)

Para entender cómo replicar esto tú solo, piensa en el flujo de datos como un ciclo de 4 pasos: **Conexión -> Escucha -> Acción -> Limpieza**.

A. La Conexión (El atributo data-key)

El desafío principal es: *"¿Cómo sabe JavaScript que cuando presiono la tecla física **'A'**, debe sonar el audio **'clap'** y iluminarse el **primer cuadro**?"*

La solución es usar un identificador común.

- Cada tecla del teclado tiene un código numérico (ASCII). La 'A' es el **65**.
- En el HTML, le pusimos una etiqueta especial a la caja visual: .
- En el HTML, le pusimos la misma etiqueta al audio: <audio data-key="65">.
- **Lógica:** JavaScript usa ese número como puente para conectar el mundo físico con el virtual.

B. La Escucha (addEventListener)

El programa necesita estar atento.

JavaScript

```
window.addEventListener('keydown', playSound)
```

- **window**: Significa que escuchamos en toda la ventana del navegador, no importa dónde hagas clic.
- **keydown**: El evento se dispara en el momento exacto en que la tecla baja.
- **Resultado**: Al bajar la tecla, se llama a la función playSound.

C. La Acción (playSound)

Esta función hace todo el trabajo pesado.

1. **Selección:** document.querySelector(...). Busca en el HTML el audio y el div que tengan el data-key igual a la tecla que presionaste.
2. **Validación:** if(!audio) return;. Si presionas una tecla que no hemos programado (como la 'Z'), audio será null (vacío). El return detiene la función inmediatamente para que no dé errores.

3. **Rebobinado:** audio.currentTime = 0. Esto es un truco importante. Si no pones esto y presionas la tecla muchas veces rápido, el navegador esperará a que el audio termine antes de volver a sonarlo. Al ponerlo en 0, lo fuerzas a reiniciar, permitiendo un efecto de batería rápida (drrrrrrum!).
4. **Estilo:** key.classList.add('playing'). Aquí JS habla con CSS. Le pone la clase .playing al elemento. El navegador ve esa clase y aplica los estilos de borde rojo y escala grande.

D. La Limpieza (removeTransition)

Si solo hiciéramos lo anterior, la tecla se quedaría roja y grande para siempre. Necesitamos quitar la clase .playing.

- **El error común:** Usar un temporizador (setTimeout) para quitar la clase después de 0.07 segundos.
- **La forma profesional:** Usar transitionend.
    - Le decimos a cada tecla: *"Avísame cuando termines tu transformación CSS"*.
    - Cuando la animación termina, JS ejecuta removeTransition.
    - Esta función simplemente hace this.classList.remove('playing'), devolviendo la tecla a su estado original.

---

3. ¿Cómo hacer esto tú solo la próxima vez?

Si quieres crear algo similar sin mirar un tutorial, sigue estos pasos mentales:

1. **Estructura (HTML):** Crea los elementos en pantalla. Asocia cada elemento con su contraparte lógica (usando data-attributes o id).
2. **Estilo (CSS):** Diseña el estado "normal" y crea una clase separada para el estado "activo" (ej: .playing, .active, .encendido).
3. **Interacción (JS):**
    - Crea un *Listener* que detecte la acción del usuario (teclado, click, mouseover).
    - Dentro de la función, selecciona el elemento correcto.
    - Modifica el elemento (reproduce sonido, cambia texto, etc.).

    - Añade la clase CSS de estado "activo".
4. **Restauración (JS):** Decide cómo volver al estado normal. ¿Por tiempo? ¿Al soltar el click? ¿Al terminar la animación (transitionend)?