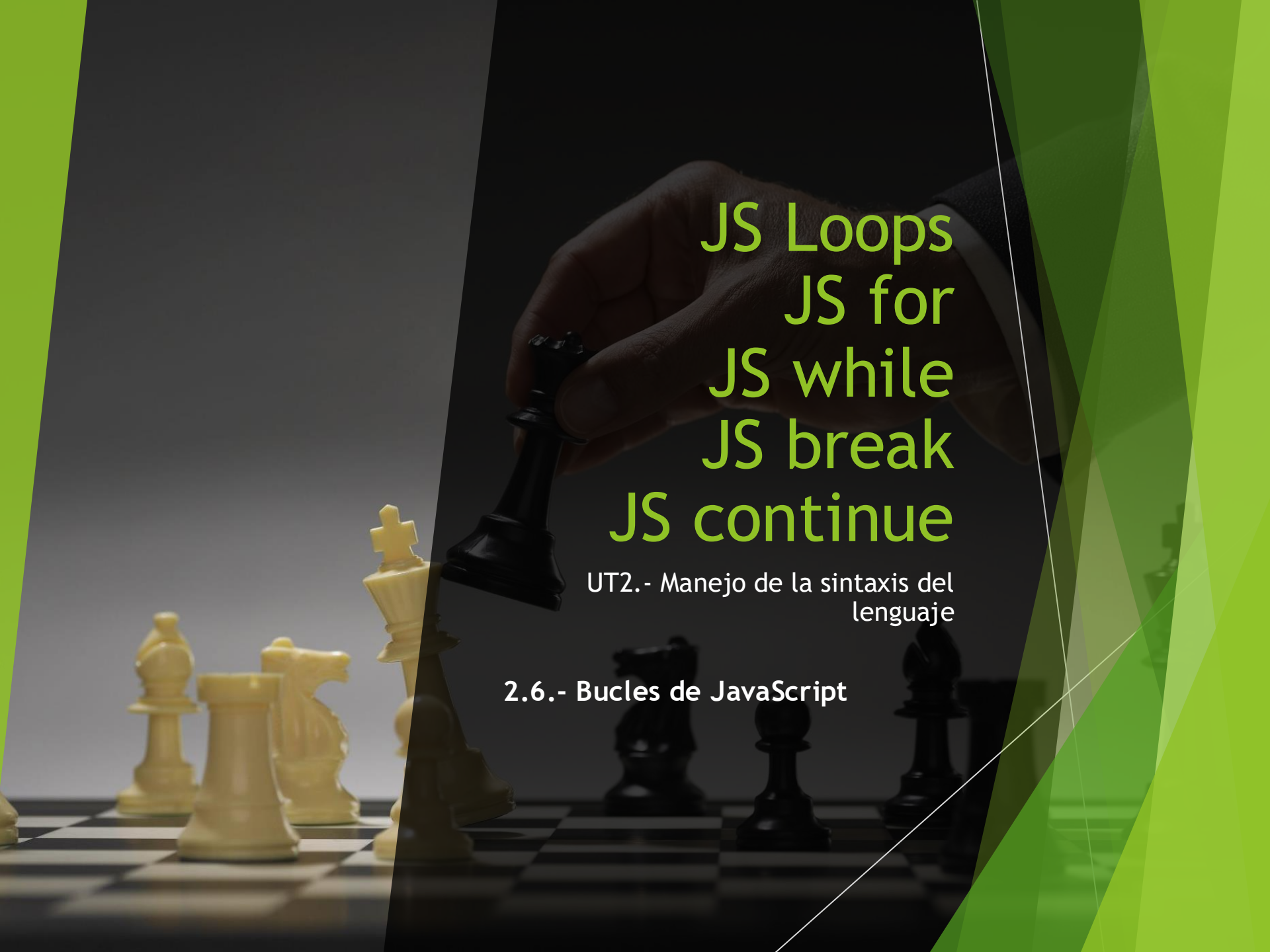


UT2.- Manejo de la sintaxis del lenguaje

## 2.6.- Bucles de JavaScript con ejemplos de Ajedrez

Desarrollo Web Entorno Cliente (DWEC)





# JS Loops JS for JS while JS break JS continue

UT2.- Manejo de la sintaxis del  
lenguaje

## 2.6.- Bucles de JavaScript



# JS Loops

UT2.- Manejo de la sintaxis del lenguaje

## 2.6.- Bucles de JavaScript con ejemplos de Ajedrez



## 2.6.- Bucles de JavaScript

### - JS Loops

#### Loop o bucle

- Sirve para repetir un bloque de código múltiples veces.

#### Tipos principales de bucles en JS

- `for` — cuando sabes cuántas repeticiones.
- `while` — se repite mientras se cumpla una condición.
- `do...while` — ejecuta al menos una vez antes de comprobar la condición.
- También: `for...in`, `for...of` para iterar objetos o colecciones.





## 2.6.- Bucles de JavaScript

### - JS Loops Bucle for

JavaScript



```
1 const piezas = ["♔", "♖", "♘", "♙", "♚"];
2 for (let i = 0; i < piezas.length; i++) {
3   console.log(`Pieza ${i + 1}: ${piezas[i]}`);
4 }
```

- ▶ Este bucle recorre el array piezas y muestra cada símbolo con su índice+1.
- ▶ Usa let para el índice, y es buena práctica que controles los límites del el bucle según length.



Mejores prácticas para que se ejecute más rápido





## 2.6.- Bucles de JavaScript

### - JS Loops Scope en bucle

- ▶ En JavaScript, el **scope** (alcance) de una variable depende de cómo se declare dentro del bucle.
  - Con var: la variable tiene **scope de función** o global → puede “escaparse” del bucle.
  - Con let o const: la variable tiene **scope de**

```
1 // Usando let (buena práctica)
2 for (let j = 0; j < 3; j++) {
3   console.log("Iteración con let:", j);
4 }
5 console.log("Acceso a j fuera del bucle:", j);
6 // ✗ Error: j is not defined
```



Ejemplo para entender mejor  
Loop Scope



## 2.6.- Bucles de JavaScript

### - Buenas prácticas con scope en bucles

- ▶ Siempre usar `let` o `const` para índices y variables de control en bucles.
- ▶ Así se evitan errores de “fugas de variables” y se respeta el principio de encapsulación.
- ▶ Usar `const` si no se va a reasignar dentro del bucle (por ejemplo, al iterar con `for...of`).



## 2.6.- Bucles de JavaScript

### - JS Loops Bucle while

JavaScript



```
1 let movimientosRestantes = 5;  
2 while (movimientosRestantes > 0) {  
3   console.log(`Puedes hacer ${movimientosRestantes} movimientos más`);  
4   movimientosRestantes--;  
5 }
```

- ▶ Mientras movimientosRestantes sea mayor que 0, ejecuta el bloque.
- ▶ Es importante decrementar la variable para evitar bucle infinito.







## 2.6.- Bucles de JavaScript

### - JS Loops Bucle do...while

JavaScript



```
1 let intento = 0;  
2 const maxIntentos = 3;  
3 do {  
4     intento++;  
5     console.log(`Intento ${intento} para mover pieza`);  
6 } while (intento < maxIntentos);
```

- Garantiza que el bloque se ejecute al menos una vez, incluso si la condición es falsa inicialmente.





## 2.6.- Bucles de JavaScript

### - JS Loops Bucle for...in para objetos

JavaScript



```
1 const tablero = {
2   a1: "♔", a2: null,
3   b1: "♞", b2: "♠"
4 };
5 for (const casilla in tablero) {
6   console.log(`En ${casilla} hay: ${tablero[casilla] ?? "casilla vacía"}`);
7 }
```

- ▶ for...in itera sobre las propiedades (claves) del objeto tablero.
- ▶ Uso de ?? para manejar valores null o undefined.





## 2.6.- Bucles de JavaScript

### - JS Loops Bucle for...of para iterables



```
1 const movimientos = ["e4", "d4", "Nf3", "c5"];
2 for (const jugada of movimientos) {
3   console.log(`Movimiento: ${jugada}`);
4 }
```

- ▶ for...of recorre directamente los valores de un iterable (array, string, etc.).
- ▶ Muy útil para listas de jugadas, secuencias de movimientos, etc.





## 2.6.- Bucles de JavaScript

### - JS Loops - Uso de break y continue en bucles



```
1 // Ejemplo con break
2 const piezas = ["♔", "♚", "♜", "♞"];
3 for (let i = 0; i < piezas.length; i++) {
4   if (piezas[i] === "♜") {
5     console.log("Encontrado alfil, saliendo del bucle");
6     break;
7   }
8   console.log(piezas[i]);
9 }
10
11 // Ejemplo con continue
12 for (const movimiento of ["e4", "d4", "0-0", "c5"]) {
13   if (movimiento === "0-0") {
14     // no listar el enroque en este contexto
15     continue;
16   }
17   console.log(`Movimiento válido: ${movimiento}`);
18 }
```

- ▶ **break** acaba el bucle inmediatamente.
- ▶ **continue** salta la iteración actual y avanza a la siguiente.



## 2.6.- Bucles de JavaScript

### - Buenas prácticas en bucles

- ▶ Evita bucles infinitos: asegúrate de modificar la variable de control.
- ▶ Usa `let` o `const`, no `var`, dentro de bucles.
- ▶ Simplifica bucles largos usando funciones auxiliares.
- ▶ Para arrays/iterables, prefiere `for...of` o `forEach` cuando el orden y la claridad lo requieran.



¿Alguna pregunta?



UT2.- Manejo de la  
sintaxis del lenguaje

## 2.6.- Bucles de JavaScript




# Aula Virtual: Actividad 08

UT2.- Manejo de la sintaxis del lenguaje

2.6.- Bucles de JavaScript



A hand is shown moving a black chess piece on a chessboard. The chessboard is in the foreground, with several white and black pieces visible. The background is dark, and there are green geometric shapes on the right side of the image. The text is overlaid on the right side of the image.

# JS Loops JS for JS while JS break JS continue

UT2.- Manejo de la sintaxis del  
lenguaje

2.6.- Bucles de JavaScript