BUCLES / LOOPS

BUCLES

SALTOS

BUCLES INFINITOS

BUCLES

EL CUENTO DE LA DIRECTORA

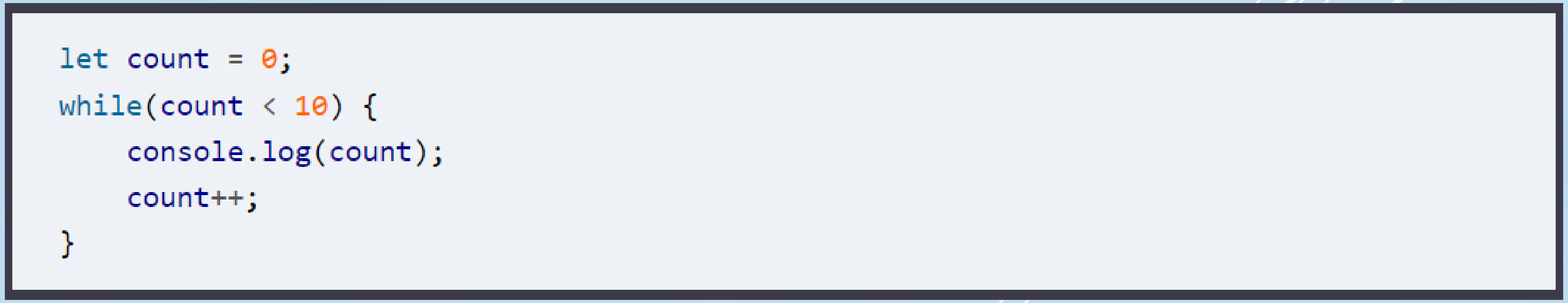
BUCLES

Bucle diario: ¿En qué momentos del día se realiza un bucle? Pensad cinco por persona.

BUCLES

while

- Se ejecuta mientras se satisfaga una condición

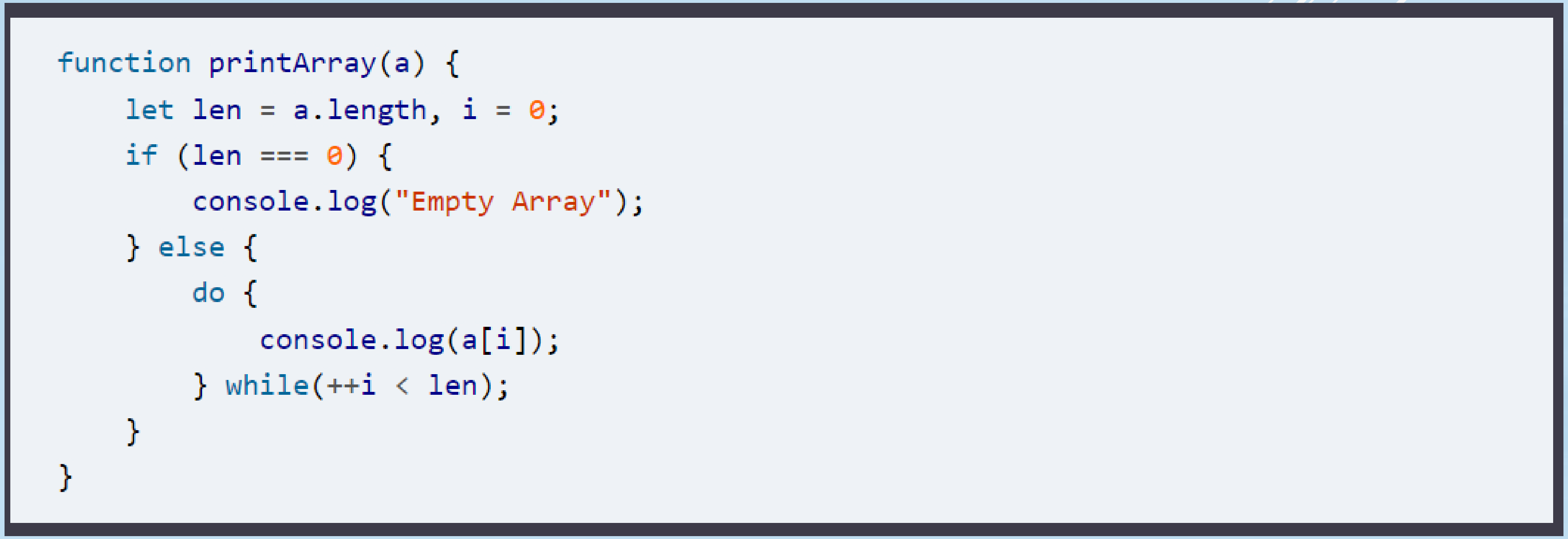


```
let count = 0;
while(count < 10) {
    console.log(count);
    count++;
}
```

BUCLES

do… while

- Mínimo se ejecuta una vez

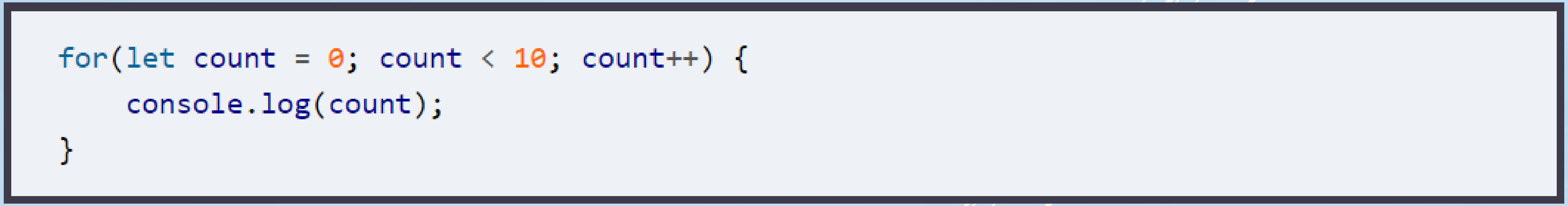


```
function printArray(a) {
    let len = a.length, i = 0;
    if (len === 0) {
        console.log("Empty Array");
    } else {
        do {
            console.log(a[i]);
        } while(++i < len);
    }
}
```

BUCLES

for

- Se ejecuta una variable incremental



```
for(let count = 0; count < 10; count++) {
    console.log(count);
}
```

BUCLES

for … of

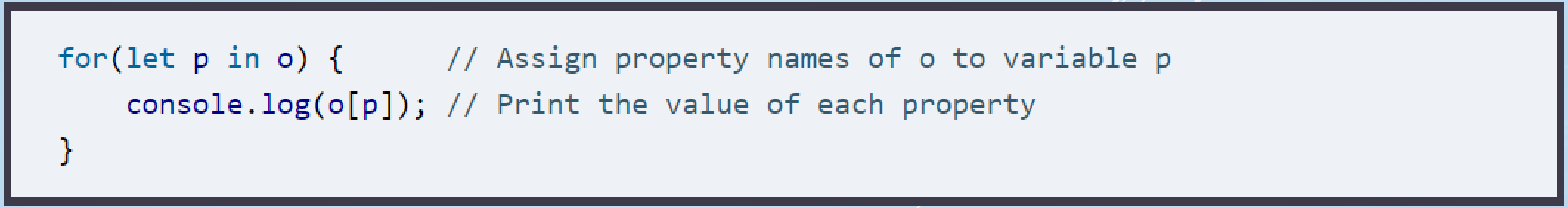
- Se ejecuta sobre un conjunto de datos (lista…), siempre que sea iterable

```
let data = [1, 2, 3, 4, 5, 6, 7, 8, 9], sum = 0;
for(let element of data) {
    sum += element;
}
sum       // => 45
```

BUCLES

for … in

- Se ejecuta sobre las propiedades de un objeto



```
for(let p in o) {      // Assign property names of o to variable p
    console.log(o[p]); // Print the value of each property
}
```

SALTOS

break

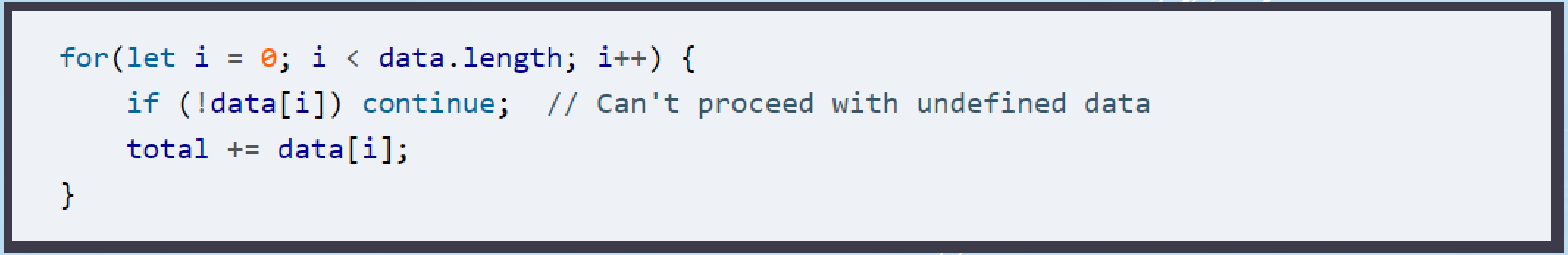
Sirve para salir definitivamente de un bucle y switch de manera interrumpida (sin satisfacer la condición o el supuesto inicial)

```
for(let i = 0; i < a.length; i++) {
    if (a[i] === target) break;
}
```

SALTOS

continue

Sirve para saltarse al inicio del bucle desde el punto en el que se ha puesto, sin seguir con lo que queda de Código dentro del mismo

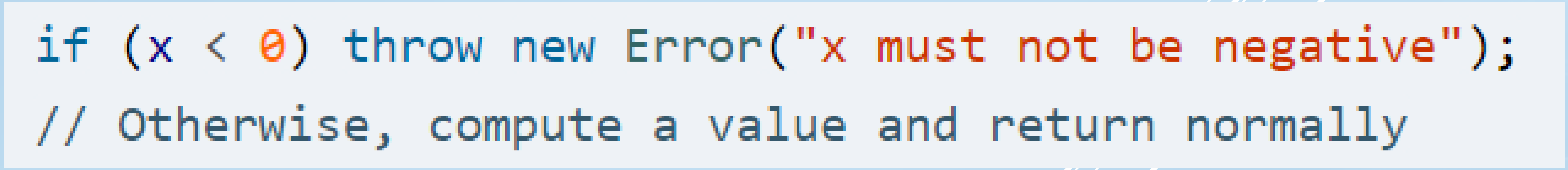


```
for(let i = 0; i < data.length; i++) {
    if (!data[i]) continue;  // Can't proceed with undefined data
    total += data[i];
}
```

SALTOS

throw

Sirve para salirse del programa, lanzando una excepción.

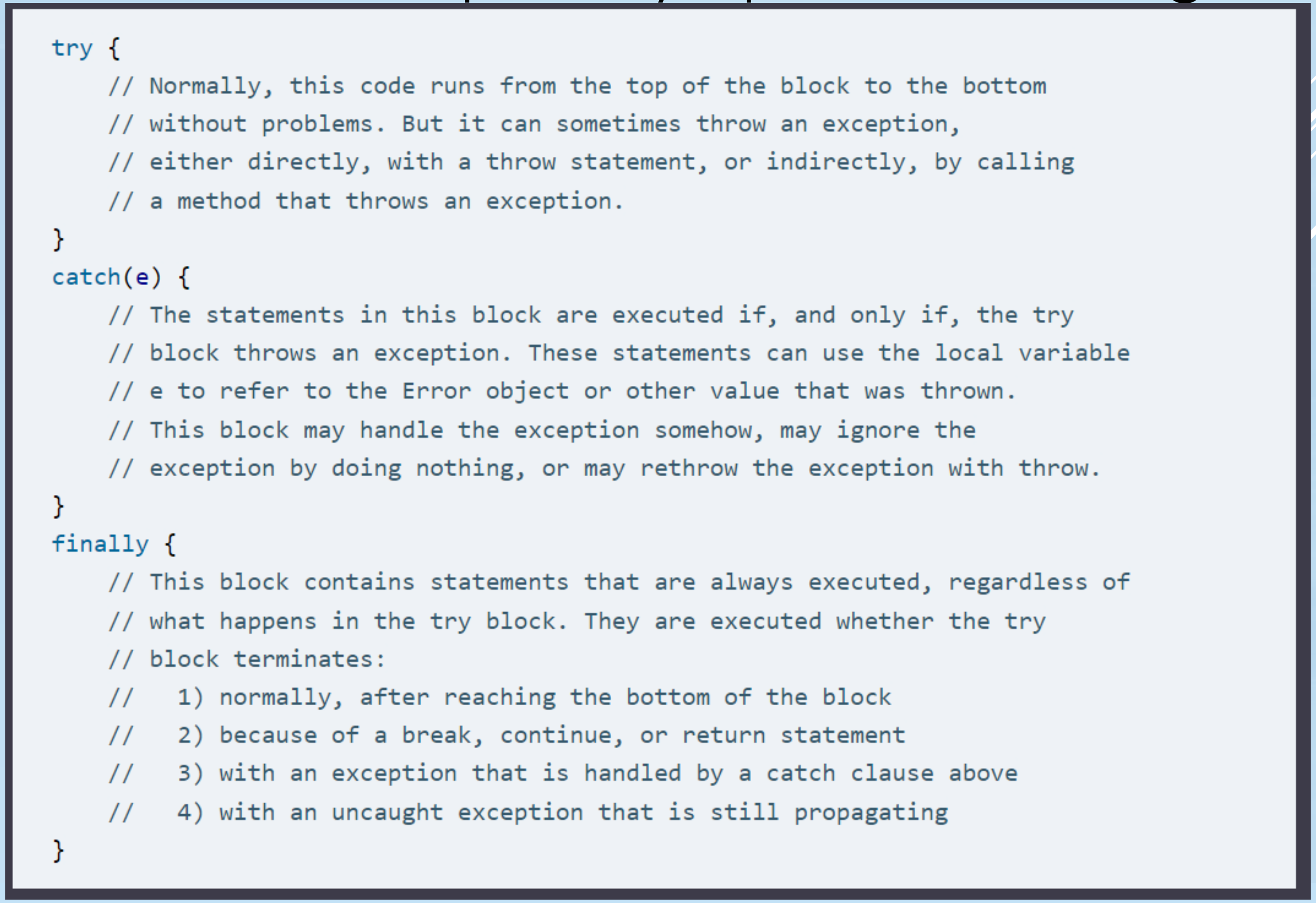


```
if (x < 0) throw new Error("x must not be negative");
// Otherwise, compute a value and return normally
```

SALTOS

try/catch/finally

Sirve para controlar una excepción y aplicar un Código dependiente de ella.



```
try {
    // Normally, this code runs from the top of the block to the bottom
    // without problems. But it can sometimes throw an exception,
    // either directly, with a throw statement, or indirectly, by calling
    // a method that throws an exception.
}
catch(e) {
    // The statements in this block are executed if, and only if, the try
    // block throws an exception. These statements can use the local variable
    // e to refer to the Error object or other value that was thrown.
    // This block may handle the exception somehow, may ignore the
    // exception by doing nothing, or may rethrow the exception with throw.
}
finally {
    // This block contains statements that are always executed, regardless of
    // what happens in the try block. They are executed whether the try
    // block terminates:
    //   1) normally, after reaching the bottom of the block
    //   2) because of a break, continue, or return statement
    //   3) with an exception that is handled by a catch clause above
    //   4) with an uncaught exception that is still propagating
}
```

BUCLES INFINITOS

¿Cuándo es útil un bucle infinito?

- Un Sistema operativo
- Un servidor
- Videojuegos
- Industria: programa que recibe inputs y genera outputs

BIBLIOGRAFÍA

1. JavaScript: The Definitive Guide, 7th Edition
2. Eloquent JavaScript, 3th edition, Marijn Haverbeke
3. Clean Code: A Handbook of Agile Software Craftsmanship (Robert C. Martin Series)
4. Modern c++ Programming with Test-Driven Development, Jeff Langr
5. Refactoring: Improve the design of existing Code, Martin Fowler
6. Game programming patterns, Robert Nystrom