M02 - JavaScript Fundamentals

•••

Loops

LOOPS

- Sometimes, certain instructions require repeated execution
- Loops are the ideal way to reproduce this effect
- A loop represents a set of instructions that must be repeated
- In the context of a loop, a repetition is referred to as an iteration
- Loop types:
  - while - the condition is checked before each iteration
  - do...while - the condition is checked after each iteration
  - for(;;) - the condition is checked before each iteration, additional settings available.

Loops

1. WHILE statement

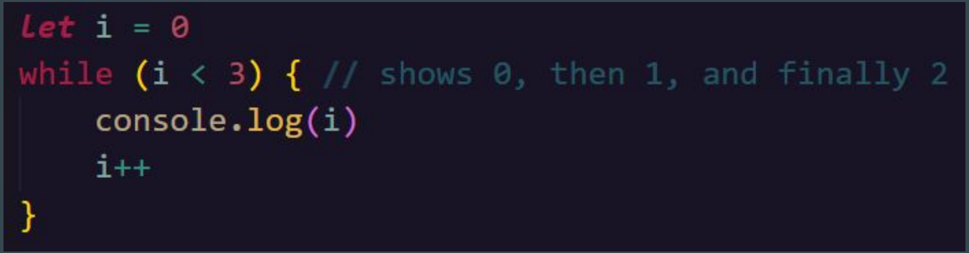
Loops

1. WHILE statement

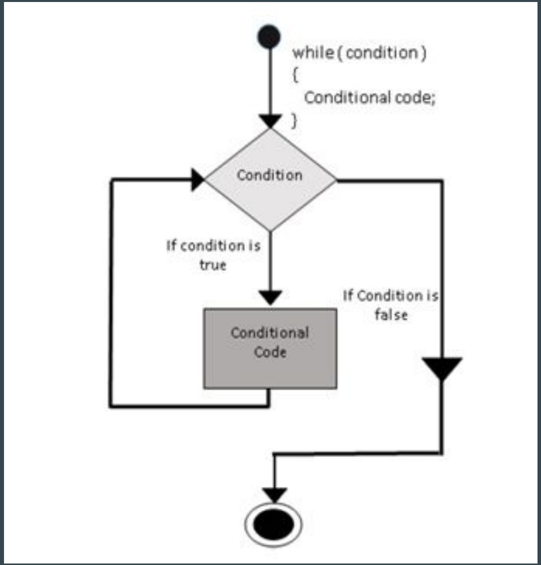
- The while loop has the following syntax:

  while(condition) {... loop body ...}

- As long as the condition is true, the loop body is executed
- For example, the cycle below shows i while i<3:



```
let i = 0
while (i < 3) { // shows 0, then 1, and finally 2
    console.log(i)
    i++
}
```



Loops

2. DO...WHILE statement

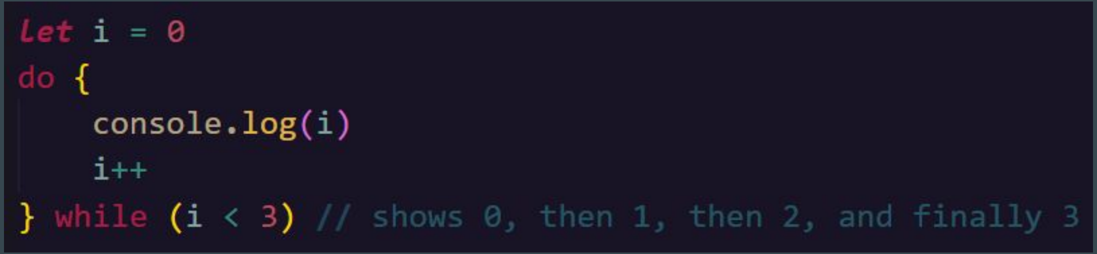
Loops

2. DO...WHILE statement

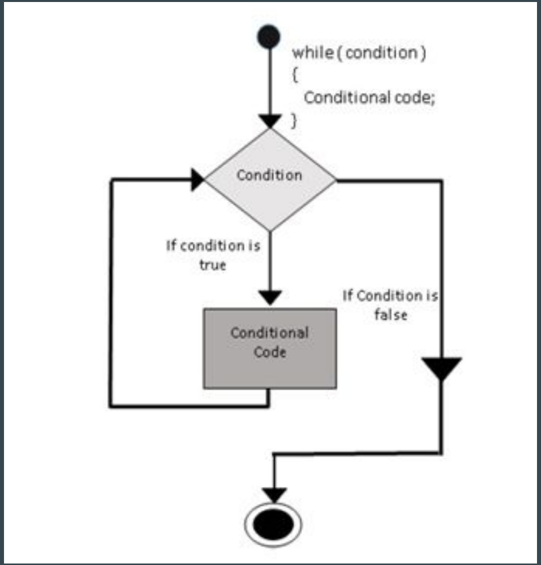
- The condition check can be moved below the loop body using the syntax:

  do {... loop body ...}while(condition)

- The loop first executes the body and then checks the condition. As long as it is true, it runs again.



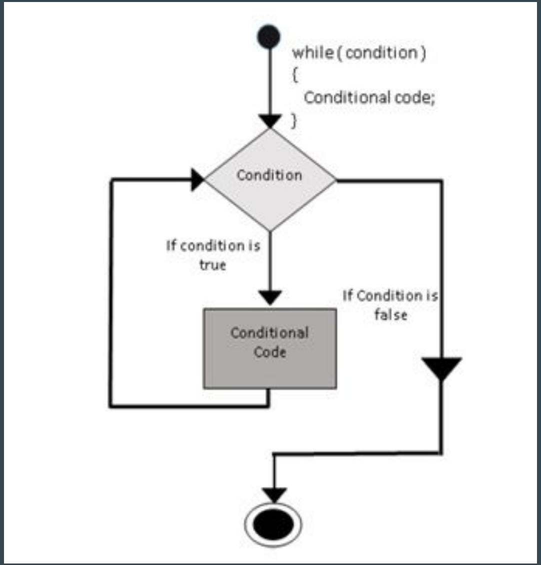
```
let i = 0
do {
    console.log(i)
    i++
} while (i < 3) // shows 0, then 1, then 2, and finally 3
```



Loops

2. DO..WHILE statement

- This form of syntax should only be used when you want the loop body to be executed at least once, regardless of the condition in effect. Usually, the other way is preferred: while (...) {...}



Loops

3. FOR statement

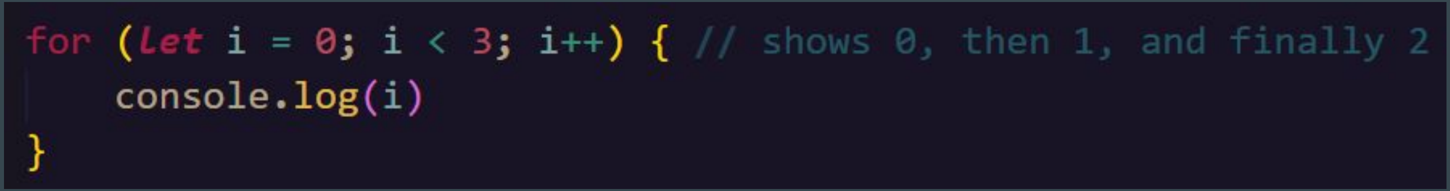
Loops

3. FOR statement

- The for loop is the most commonly used loop
- Syntax:

```
for (begin; condition; step) {
    // ... loop body ...
}
```

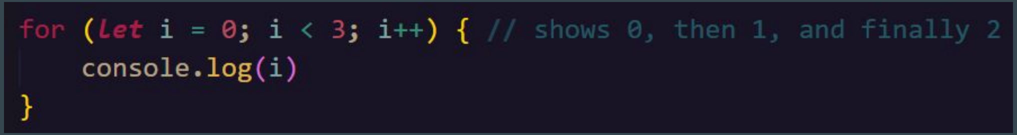
- Example:



```
for (let i = 0; i < 3; i++) { // shows 0, then 1, and finally 2
    console.log(i)
}
```

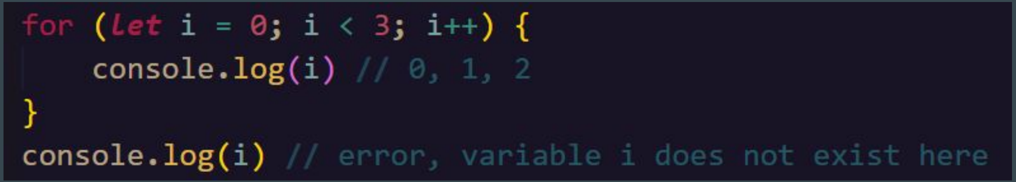
Loops

3. FOR statement



```
for (let i = 0; i < 3; i++) { // shows 0, then 1, and finally 2
    console.log(i)
}
```

- Explanation:
  - begin i = 0 executes once upon entering the loop
  - condition i < 3 checked before each iteration of the cycle. If false, the cycle stops.
  - step i++ executes after the body in each iteration, but before checking the condition.
  - body console.log(i) runs repeatedly while condition is true
-
- Use of inline variable
  - Variable i only exists within the block where it was defined

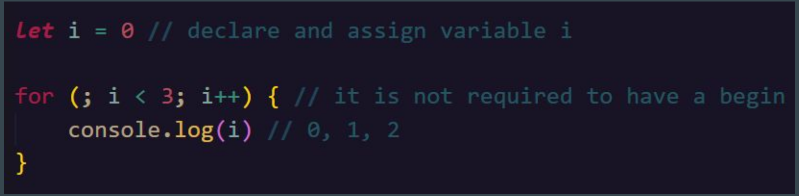


```
for (let i = 0; i < 3; i++) {
    console.log(i) // 0, 1, 2
}
console.log(i) // error, variable i does not exist here
```

Loops

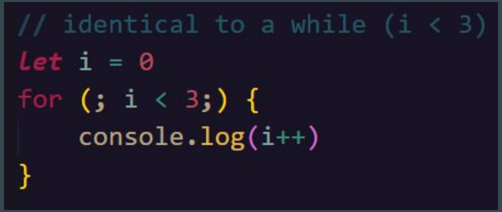
3. FOR statement

- Skip parts
  - Any part of the for cycle can be ignored
  - Remove the begin



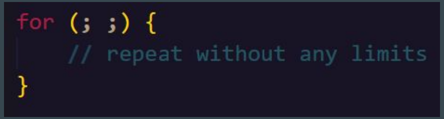
```
let i = 0 // declare and assign variable i

for (; i < 3; i++) { // it is not required to have a begin
    console.log(i) // 0, 1, 2
}
```

  - Remove the step



```
// identical to a while (i < 3)
let i = 0
for (; i < 3;) {
    console.log(i++)
}
```

  - Remove all

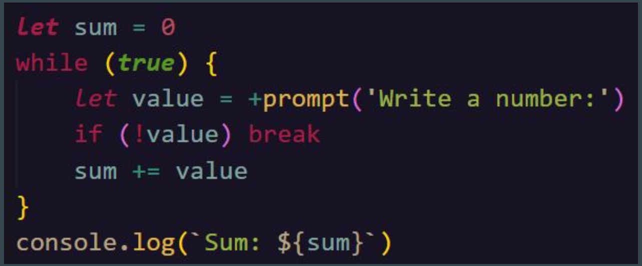


```
for (; ;) {
    // repeat without any limits
}
```

Loops

3. FOR statement

- Loop break
  - Normally, a cycle ends when its condition becomes false
  - We can force the exit at any time using the special interrupt directive: break



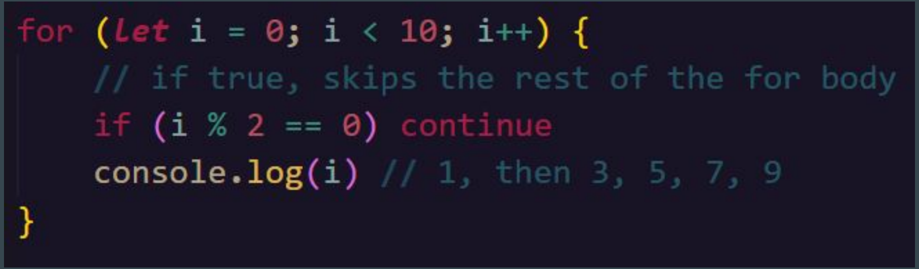
```
let sum = 0
while (true) {
    let value = +prompt('Write a number:')
    if (!value) break
    sum += value
}
console.log(`Sum: ${sum}`)
```

  - The combination of infinite cycle + break is great for situations where the condition of a loop must be checked not at the beginning or end of the cycle, but in the middle or even at various places in the loop body

Loops

3. FOR statement

- Skip to the next iteration
  - The continue directive is a lighter version of the break. Not for the whole cycle. Instead, it interrupts the current iteration and forces the loop to start a new one (if the condition allows).
  - We can use it if we finish the current iteration and want to move on to the next one
  - Example:



```
for (let i = 0; i < 10; i++) {
    // if true, skips the rest of the for body
    if (i % 2 == 0) continue
    console.log(i) // 1, then 3, 5, 7, 9
}
```