

# MIND HUB.

# LOOPS



# Index

→ **1** For

→ **2** For in

→ **3** While

→ **4** Do While



# 'for' loop

We often need to perform similar actions many times in a row.

For example, when we need to print products from a list one after the other. Or simply run the same code for each number from 1 to 10.

**Loops are a way of repeating the same part of the code several times.**

The **for** loop is the most commonly used.

```
for(initialization, condition, step){  
  
    //..loop body...  
  
}
```

A single execution of the loop body is called an *iteration*.

# 'for' loop explained

Parts of a 'for' loop:

*initialization, condition, step, body.*

Let's learn the meaning of these parts with an example.

The loop on the right executes **console.log(i)** for i from 0 up to (but not including) 3:

```
for( let i = 0; i < 3; i = i + 1 ){  
    console.log( i )  
}
```

Let's examine the for statement part by part:

part	code	action
<i>initialization</i>	<b>let i = 0</b>	It is <b>executed once</b> when entering the loop.
<i>condition</i>	<b>i &lt; 3</b>	Checked before each iteration of the loop, <b>if it fails the loop stops.</b>
<i>step</i>	<b>i = i + 1 (or i++)</b>	It is executed after the body in each iteration, but before the condition check.
<i>body</i>	<b>console.log(i)</b>	Run again and again as long as the condition is true.

# 「for」 loop example

If you are new to loops, then it may help to go back to the example and reproduce **how it works step by step on a piece of paper**.

*This is exactly what happens in our case (step by step):*

```
// run initialization
let i = 0
// if condition → run body and run step
if( i < 3 ) {
  console.log( i )
  i = i + 1
}
// if condition → run body and run step. i value is now 1
if( i < 3 ) {
  console.log( i )
  i = i + 1
}
// if condition → run body and run step. i value is now 2
if( i < 3 ) {
  console.log( i )
  i = i + 1
}
// ... finish, because now i = 3
```

# 'for in' loop

It is built to iterate **object properties**.

For example, when we need to print properties of an object in a list one after another.

**We do not use only for since the objects do not have an index to be traversed or iterated.**

The **for in** loop is not the most used but sometimes it will be very useful to understand it.

```
for (props in object){  
  
    instruction;  
  
}
```

Do not forget the **;** otherwise the following **instruction** line will be taken as the instruction to be executed.

# 'for in' loop explained

Parts of a 'for' loop:

*props, object, instruction*

Let's learn the meaning of these parts with an example. The loop on the right runs **console.log()** to go through an object and get its properties, sorting its values based on the property

```
for( let props in object){  
    console.log( `${props}: ${object[props]}` )  
}
```

Let us examine the statement part by part:

part	code	action
<i>props</i>	<b>props</b>	Represents the properties of the object.
<i>object</i>	<b>object</b>	The object to iterate.
<i>instruction</i>	<b>console.log()</b>	It is the value returned by the iteration of this object.



# 'for in' loop example

Look at this example, on the left the code in Visual Studio and on the right the output in our browser console:

```
let car = {  
  brand : "Peugeot",  
  model : "307",  
  color: "black",  
  engine: 1.8  
}  
for( let props in car ){  
  console.log( `${props}: ${car[props]}` )  
}
```

## Output

---

brand: Peugeot

---

model: 307

---

color: black

---

engine: 1.8

---

# 'while' loop

The **while** loop has the following syntax:

Let's reproduce the 'while' version of the previous example with a 'for' loop:

**As long as (while) condition is true, the loop body code is executed.**

For example, the loop on the right prints *i* as long as  $i < 3$ :

What would happen if we do not put *i++* inside the loop?

```
while (condition) {  
  // code  
  // so-called "loop body"  
}
```

```
let i = 0  
while( i < 3){  
  console.log(i)  
  i++  
}
```

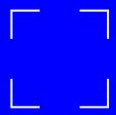
# 'do while' loop

Using the `while` structure, its content is only executed if a condition **is checked**, which can occur 0, 1 or more times.

**Do While** works in a similar way, only that we make sure that the content is executed at least once, i.e. even if **its condition is not met**, its content is executed.

```
do{  
  // code  
  // so-called "loop body"  
}while (condition)
```

```
let i = 0  
do {  
  console.log(i)  
  i++  
} while (i < 3);
```



MIND  
HUB.

