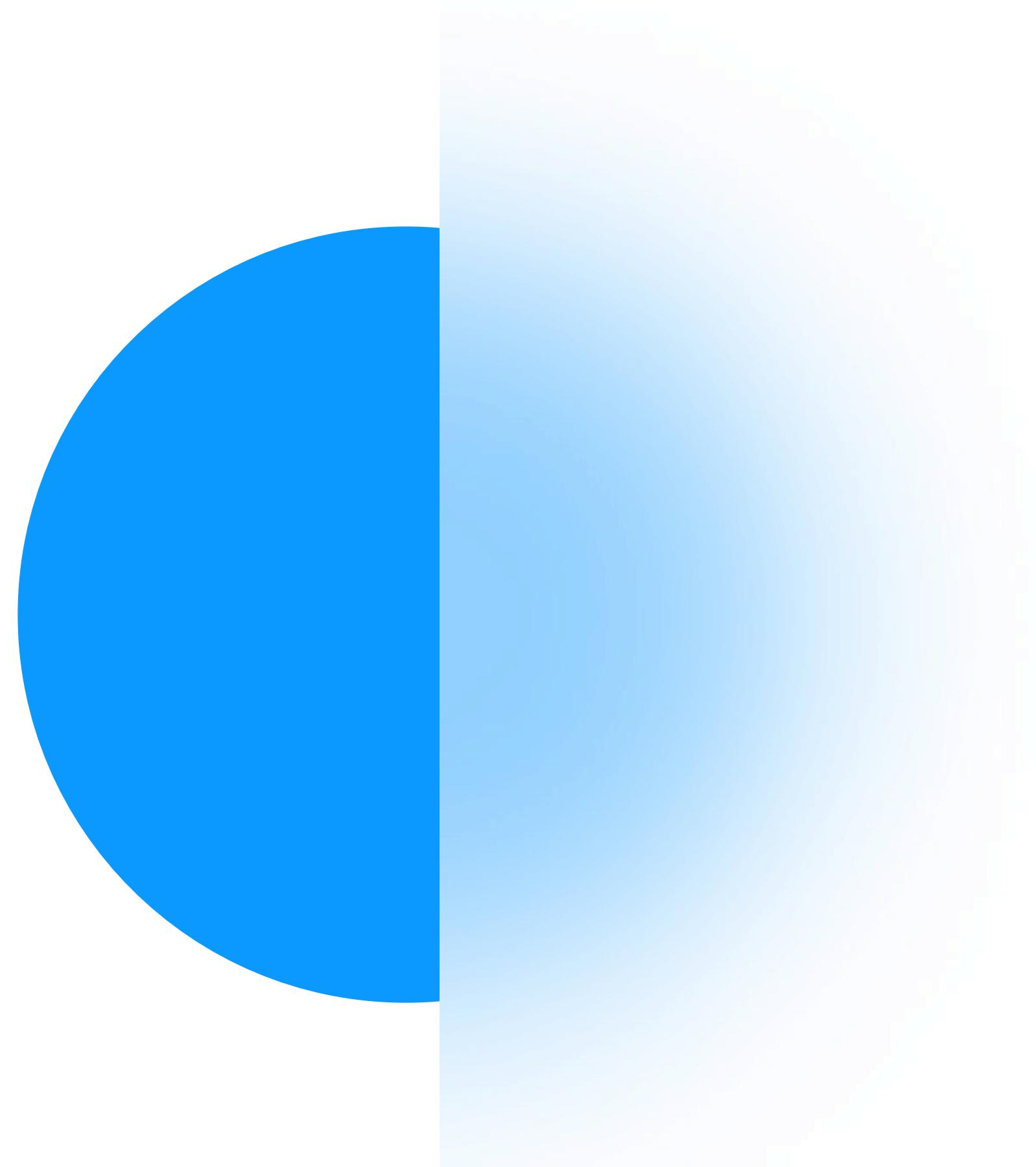


# JS ( JavaScript)

Les 2 Conditional Statements & Arrays & Loops



الموضوع	الوقت (التوقيت السوري)	التاريخ
HTML: Basics & Elements & page anatomy	12:00	2025\3\1
HTML: Basics check	12:00	2025\3\8
CSS: Basics & Selectors	12:00	2025\3\15
Typography & Colors	12:00	2025\3\22
Positions & :has()   :not()	12:00	2025\3\29
Flexbox	12:00	2025\4\5
Grid	19:30	2025\4\7
States & Media Querie	19:30	2025\4\9
Animations	19:30	2025\4\11
JS: Basics & Events & First function (html+css+js)	19:30	2025\4\14
JS: Conditional Statements & Arrays & Loops	19:30	2025\4\16
JS: API & Fetch data (pokeapi)	19:30	2025\4\18
Last check	19:30	2025\4\20

# Conditional Statements

# Conditional Statements

## عمليات المقارنة (Comparison Operators)

الوصف / النتيجة	الاسم	الرمز
true $\Rightarrow$ "5" == 5	مساواة (غير صارمة)	<code>==</code>
false $\Rightarrow$ "5" === 5	مساواة صارمة	<code>===</code>
false $\Rightarrow$ "5" != 5	عدم مساواة (غير صارمة)	<code>!=</code>
true $\Rightarrow$ "5" ==! 5	عدم مساواة صارمة	<code>==!</code>
true $\Rightarrow$ 5 < 7	أكبر من	<code>&lt;</code>
true $\Rightarrow$ 5 > 3	أصغر من	<code>&gt;</code>
true $\Rightarrow$ 5 =< 5	أكبر من أو يساوي	<code>=&lt;</code>
true $\Rightarrow$ 6 => 4	أصغر من أو يساوي	<code>=&gt;</code>

# Conditional Statements

## العبارات الشرطية (Conditional Statements)

- تُستخدم لاتخاذ قرارات في الكود بناءً على شروط معينة.
  - تساعد في تنفيذ كتل من الكود فقط عند تحقق شرط معين.
- الأنواع الأساسية:**
- if: لتنفيذ كتلة من الكود إذا تحقق شرط.
  - if...else: لتحديد مسار بديل إذا لم يتحقق الشرط.
  - if...else if...else: للتعامل مع حالات متعددة.
  - (عامل ثلاثي): طريقة مختصرة لكتابة عبارة شرطية



```
1 let score = 85;
2
3 if (score >= 90) {
4   console.log("ممتاز");
5 } else if (score >= 70) {
6   console.log("جيد جداً");
7 } else {
8   console.log("حسن");
9 }
10
11 // باستخدام عامل ثلاثي
12 let result = score >= 70 ? "راسب" : "نجاح";
13 console.log(result);
```

**ملاحظة مهمة:** لا انصح باستعمال الطريقة المختصرة للمبتدئين.



```
1 // && and
2 if (age > 1 && age < 18) {
3   console.log("You are a child");
4 } else if (age >= 18 && age < 60) {
5   console.log("You are an adult");
6 } else {
7   console.log("You are an old person");
8 }
9
10 // || or
11 if (age > 1 || age < 60) {
12   console.log("You are a child or an old person");
13 } else {
14   console.log("You are an adult");
15 }
16
17 // ! not
18 if (!age > 18) {
19   console.log("You are a child");
20 } else {
21   console.log("You are an adult");
22 }
23
24 // == equal
25 console.log(0 == "0"); // true because "0" is converted to 0
26 console.log(false == 0); // true because false is converted to 0
27 console.log(3 == 4); // false because the values are different
28
29 // === strict equal
30 console.log(0 === "0"); // false because the types are different
31 console.log(false === 0); // false because the types are different
32 console.log(0 === 0); // true because the types are the same
33 console.log(3 === 4); // false because the values are different
34
35 // != not equal
36 console.log(0 != "0"); // false because "0" is converted to 0
37
38 // !== strict not equal
39 console.log(0 !== "0"); // true because the types are different
40
```

# Conditional Statements

## Logica

المنطق هو أساس اتخاذ القرارات في الكود باستخدام قيم منطقية.

- يُساعد المنطق في اتخاذ قرارات دقيقة بناءً على شروط متعددة، مما يجعل الكود أكثر مرونة وقابلية للتوسيع.

- يمكن استخدامه في الحلقات للتحكم بتكرار الكود، وفي المصفوفات للبحث والتصفية، وفي الدوال لاتخاذ قرارات داخلية.

**ملاحظة مهمة:** == هو الخيار الآمن للمقارنة حيث لا يقوم بتحويل الأنواع، بينما = قد يؤدي إلى نتائج غير متوقعة بسبب تحويل الأنواع التلقائي.

# Arrays

# Arrays

## المصفوفات

المصفوفة هي بنية بيانات تُخزن مجموعة من القيم في متغير واحد. يمكن أن تحتوي على أنواع بيانات مختلفة (أرقام، نصوص، كائنات، إلخ).

- يبدأ ترقيم عناصر المصفوفة من الـ **0** ويسمى **index**

```
1 const numbers = [1, 2, 3, 4, 5]; // array of numbers
2 const fruits = ["apple", "banana", "cherry"]; // array of strings
3
4 // nested (متداخل) arrays
5 const nested = [
6   [1, 2],
7   [3, 4],
8   [5, 6],
9 ];
10 // array of objects
11 const objects = [
12   { name: "John", age: 18 },
13   { name: "Jak", age: 20 },
14 ];
15
16 const mixed = [1, "apple", true, { name: "John" }]; // array of mixed data types
17
18
```

# Arrays



```

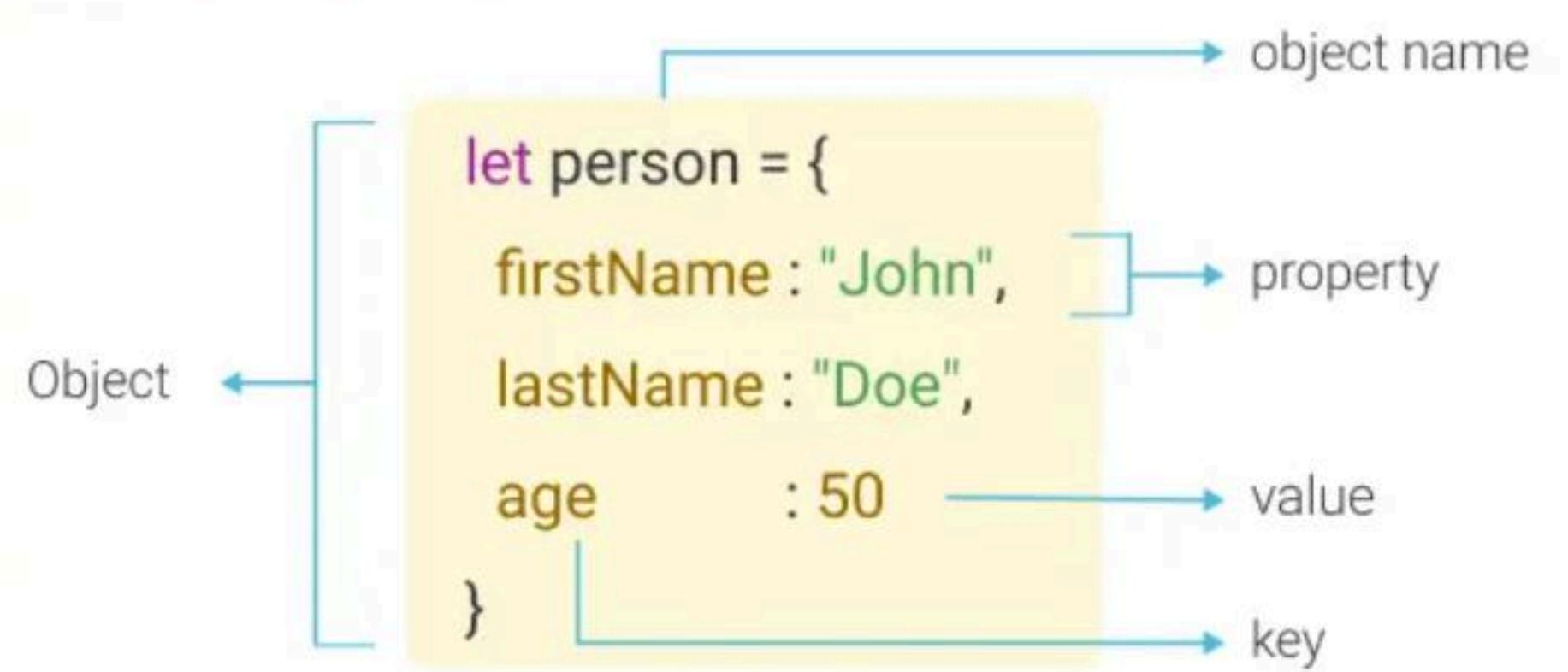
1 // get element from an array
2 const fruits = ["apple", "banana", "cherry"];
3 console.log(fruits[0]); // get the element at index 0 (apple)
4
5 // objects
6 const persons = [
7   {
8     name: "John",
9     age: 30,
10    city: "New York",
11  },
12  {
13    name: "Doe",
14    age: 20,
15    city: "New York",
16  },
17];
18 console.log(persons[0].name); // get the value of the name property (John)

```

## التعامل مع المصفوفات

- الوصول للعناصر داخل المصفوفة فيتم ذلك بناءً على رقم العنصر.
- للوصول إلى عناصر داخل object في مصفوفة فيتم ذلك عن طريق رقم key و index.

JavaScript Object Structure



# Arrays

## التعامل مع المصفوفات

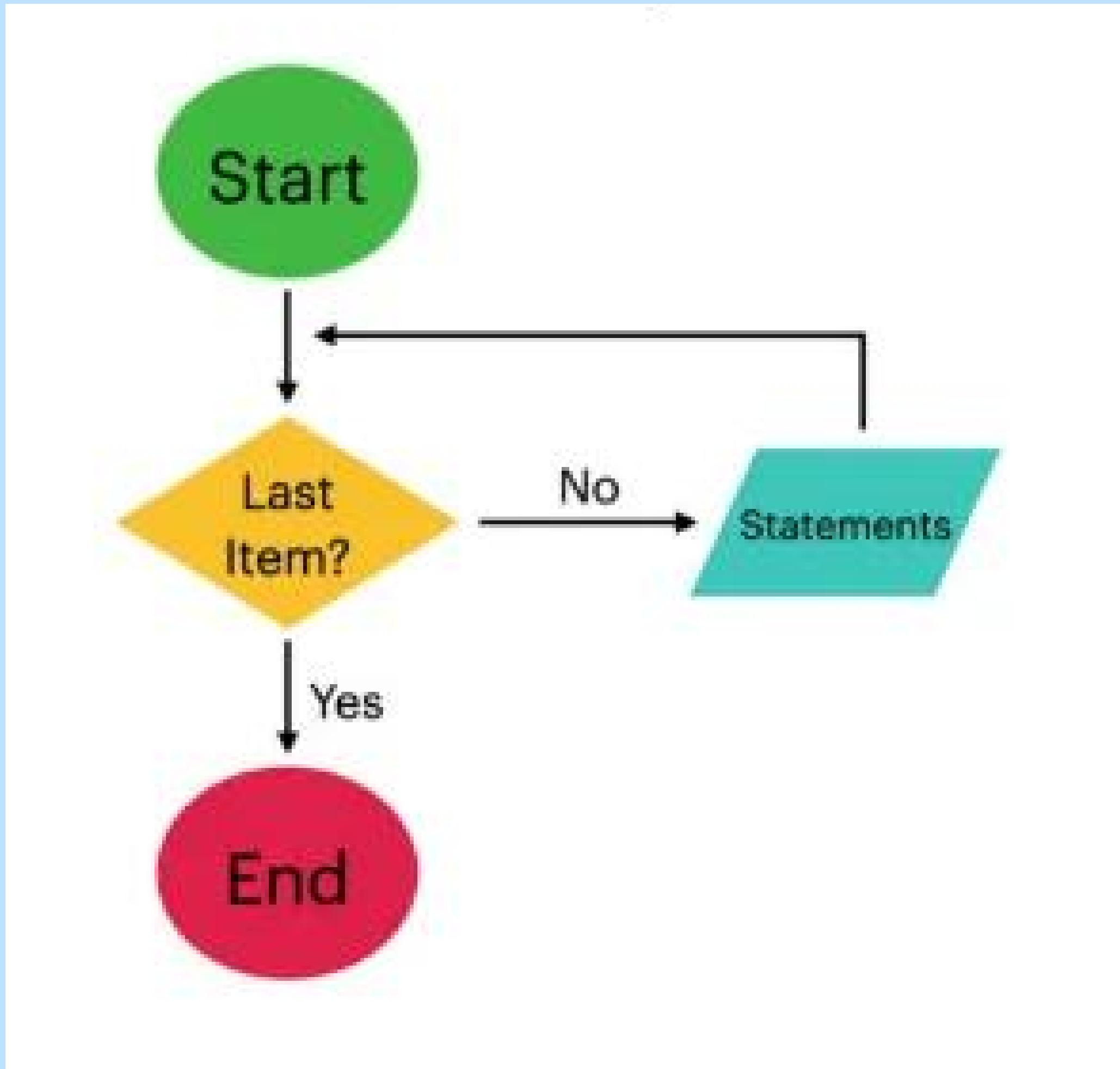
```
1 const fruits = ["apple", "banana", "cherry"];
2 fruits[0] = "pear"; // change the element at index 0 to pear
3 fruits.push("orange"); // add a new element (orange) to the end of the array
4 fruits.pop(); // remove the last element (orange) from the array
5 fruits.shift(); // remove the first element (pear) from the array
6 fruits.unshift("kiwi"); // add a new element (kiwi) to the beginning of the array
7 fruits.splice(2, 1); // remove 1 element at index 2 (cherry) from the array
8 fruits.splice(1, 2, "mango", "grape"); // remove 2 elements at index 1 (banana and cherry) and add mango and grape
9 fruits.slice(1, 3); // get a part of the array (banana, cherry) and return it as a new array
10
11
12 console.log(indexOf("banana")); // get the index of an element (1)
13 console.log(fruits.length); // get the length of the array (3)
14
15 // Advanced array methods
16 // map() - creates a new array by performing a function on each array element
17 const numbers = [1, 2, 3].map((number) => number * 2);
18 console.log(numbers); // [2, 4, 6]
19
20 // filter() - creates a new array with elements that pass a test
21 const evenNumbers = [1, 2, 3, 4, 5].filter((number) => number % 2 === 0);
22 console.log(evenNumbers); // [2, 4]
23
24 // reduce() - reduces an array to a single value
25 const sum = [1, 2, 3, 10].reduce((total, number) => total + number, 0);
26 console.log(sum); // 16
27 // total is the accumulator
28 // number is the current value
29 // 0 is the initial value of the accumulator
30 // if the initial value is 1, the result would be 17
31
32 // forEach() - calls a function for each array element
33 const colors = ["red", "green", "blue"];
34 colors.forEach((color) => console.log(color));
35
36 // find() - returns the first array element that passes a test
37 const firstEvenNumber = [1, 2, 3, 4, 5].find((number) => number % 2 === 0);
38 console.log(firstEvenNumber); // 2
39
40 // some() - checks if any array element passes a test
41 const hasEvenNumber = [1, 2, 3, 4, 5].some((number) => number % 2 === 0);
42 console.log(hasEvenNumber); // true
```

# Loops

# Loops

## الحلقات:

- تُستخدم لتكرار تنفيذ كود معين عدة مرات.
- تساعد على التعامل مع المصفوفات والمجموعات الكبيرة من البيانات بطريقة مبسطة.



# Loops

## for of loop

هو يبسط التكرار على الكائنات القابلة للتكرار مثل المصفوفات.



```
1 // for of loop
2 const arr = [1, 2, 3, 4, 5];
3 for (const item of arr) {
4     console.log(item);
5 }
6 // resulte: 1, 2, 3, 4, 5
```

## for loop

تُستخدم عندما تعرف عدد مرات التكرار.

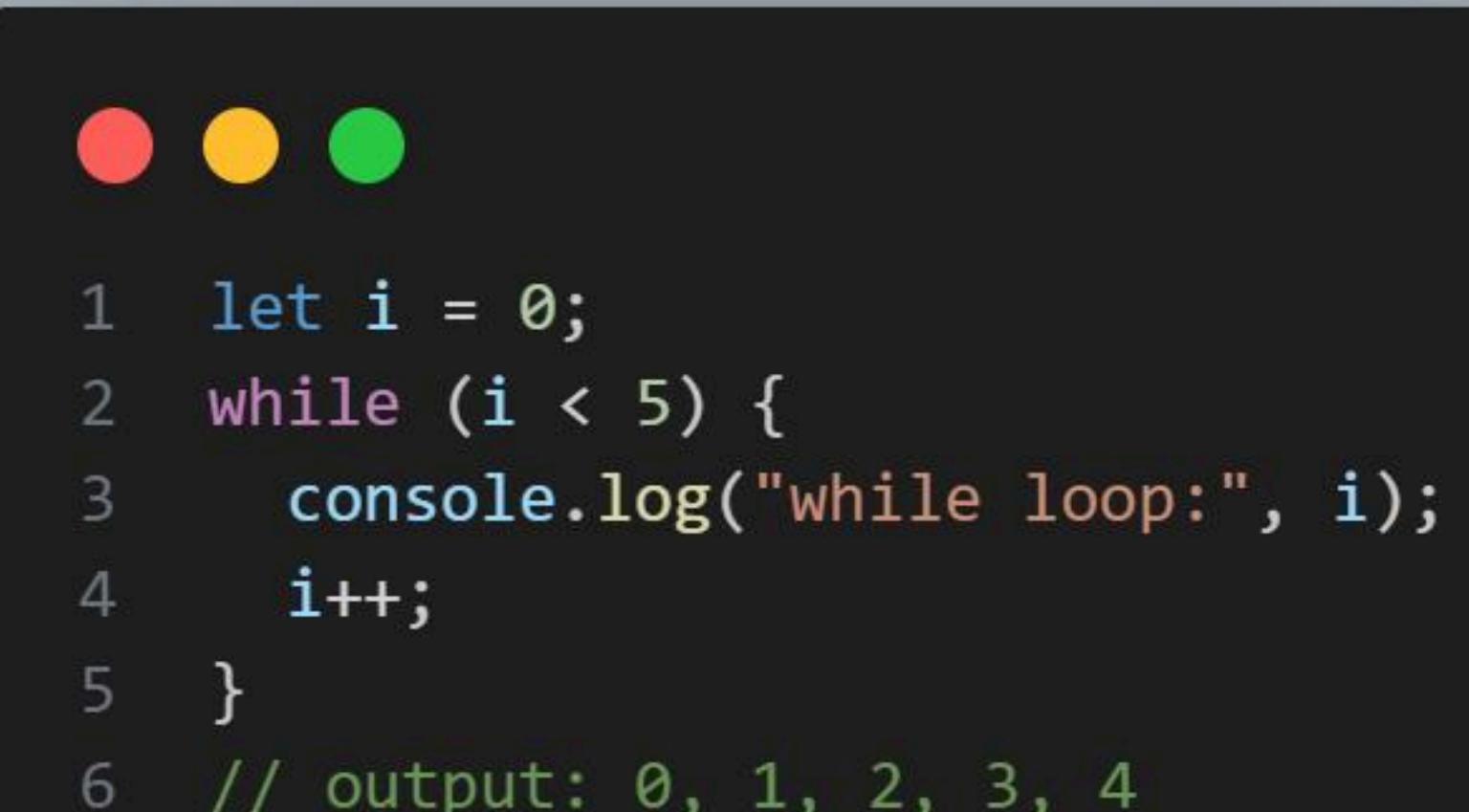


```
1 // for loop
2 for (let i = 0; i < 5; i++) {
3     console.log(dataResults[i]);
4 }
5 // resulte: 1 2 3 4 5
6
7 // forl loop with array
8 const data = ["a", "b", "c", "d", "e"];
9 for (let i = 0; i < data.length; i++) {
10    console.log(data[i]);
11 }
12 // resulte: a b c d e
13
```

# Loops

## while loop

يتم استخدام حلقة `while` لتنفيذ كتلة من التعليمات البرمجية طالما كان الشرط المحدد صحيحًا.

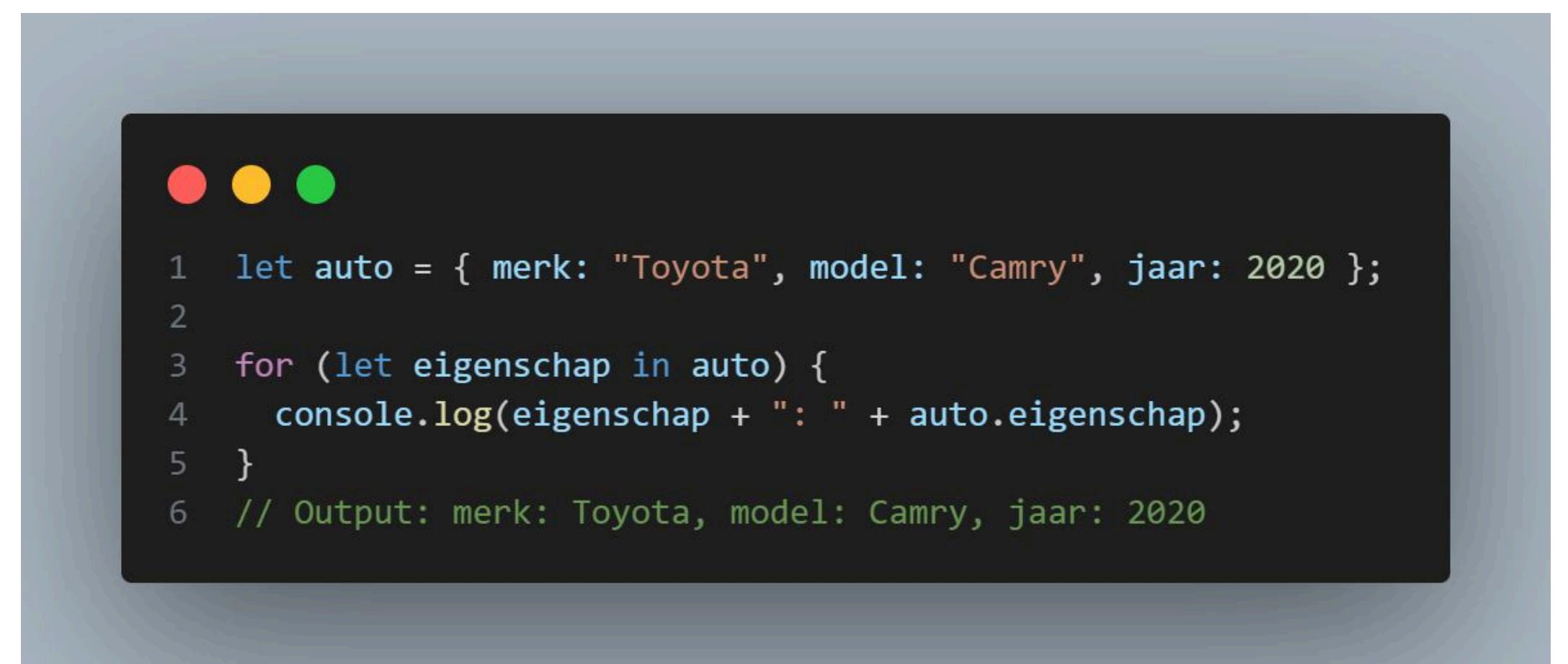


The screenshot shows a browser developer tools console with three colored dots (red, yellow, green) at the top. The code below is displayed:

```
1 let i = 0;
2 while (i < 5) {
3     console.log("while loop:", i);
4     i++;
5 }
6 // output: 0, 1, 2, 3, 4
```

## for in loop

يتم استخدام حلقة `for...in` للتكرار عبر خصائص الكائن.



The screenshot shows a browser developer tools console with three colored dots (red, yellow, green) at the top. The code below is displayed:

```
1 let auto = { merk: "Toyota", model: "Camry", jaar: 2020 };
2
3 for (let eigenschap in auto) {
4     console.log(eigenschap + ": " + auto[eigenschap]);
5 }
6 // Output: merk: Toyota, model: Camry, jaar: 2020
```

# Loops

## do while loop

تنفذ حلقة do...while كتلة من التعليمات البرمجية مرة واحدة على الأقل، حتى لو لم يكن الشرط صحيحًا.

شرط متحقق منذ البداية

```
1 let i = 0;
2 do {
3   console.log(i);
4   i++;
5 } while (i < 5);
6 // output: 0, 1, 2, 3, 4
```

شرط غير متحقق منذ البداية

```
1 let j = -1;
2 do {
3   console.log(j);
4   j++;
5 } while (j < 5 && j > 0);
6 // output: -1
```

# Loops

## forEach

تستدعي طريقة `forEach()` دالة `(function)` لكل عنصر في المصفوفة.

لا يتم تنفيذ طريقة `forEach()` للعناصر الفارغة (مصفوفة فارغة).

كتابة محتوى الدالة `(function)` فوراً بشكل سهمي

استدعاء دالة `(function)`

```
1
2 const array = [1, 2, 3, 4, 5];
3 array.forEach((item, index) => {
4   console.log(item, index);
5 });
6
7 const objectsArray = [
8   { name: "John", age: 25 },
9   { name: "Jan", age: 30 },
10 ];
11 objectsArray.forEach((item) => {
12   console.log(item.name, item.age);
13 });
```

```
1 const array = [1, 2, 3, 4, 5];
2 array.forEach(printItem);
3 function printItem(item, index) {
4   console.log(item, index);
5 }
6
7 const objectsArray = [
8   { name: "John", age: 25 },
9   { name: "Jan", age: 30 },
10 ];
11 objectsArray.forEach(printObject);
12 function printObject(item) {
13   console.log(item.name, item.age);
14 }
```

The end of JS les 2