




What is an Iterable?

An **iterable** is any object that implements the `Symbol.iterator` method. This method returns an **iterator** — an object that follows the **iterator protocol** and can be used to step through a sequence of values.



Common Iterable Objects in JavaScript

Here's a list of built-in **iterables** in JavaScript:

 Type	 Example	 Iterable
Array	<code>[1, 2, 3]</code>	Yes
String	<code>"hello"</code>	Yes
Map	<code>new Map()</code>	Yes
Set	<code>new Set()</code>	Yes
TypedArray	<code>new Uint8Array([1,2])</code>	Yes
Arguments object	(inside a function)	Yes
DOM NodeList	<code>document.querySelectorAll()</code>	Yes



Non-iterables

These are **not** iterable by default:

 Type	 Reason
Plain Object <code>{ key: 'value' }</code>	Doesn't implement <code>Symbol.iterator</code>
Number <code>42</code>	Not a collection
Boolean <code>true</code>	Not iterable

How to Check if Something is Iterable

```
const isIterable = (obj) => obj != null && typeof obj[Symbol.iterator] === 'function';

console.log(isIterable([1, 2, 3])); //  true
console.log(isIterable({ a: 1 })); //  false
```

Example: Using `for...of` with Iterables

```
const str = "hello";
for (const char of str) {
  console.log(char); // h, e, l, l, o
}

const arr = [10, 20, 30];
for (const num of arr) {
  console.log(num); // 10, 20, 30
}
```
