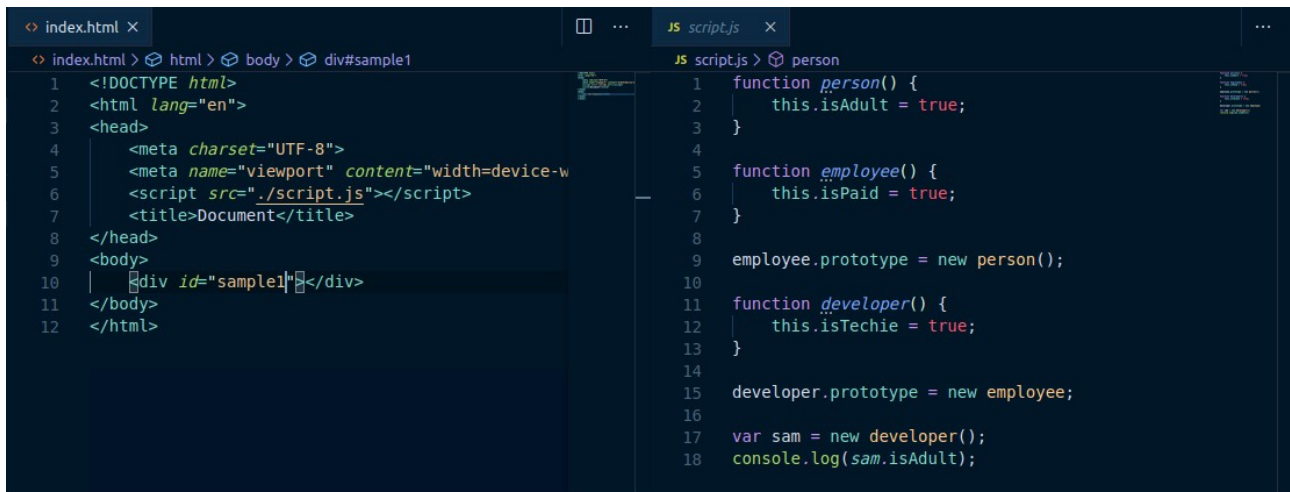


Q1. Create a hierarchy of person, employee and developers.

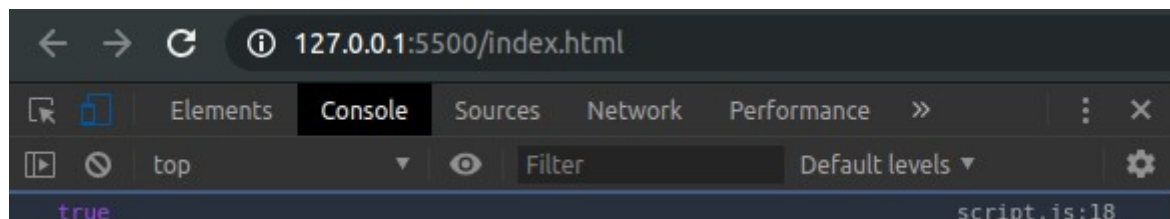


The screenshot shows a VS Code editor with two files open. The left pane shows `index.html` with the following content:

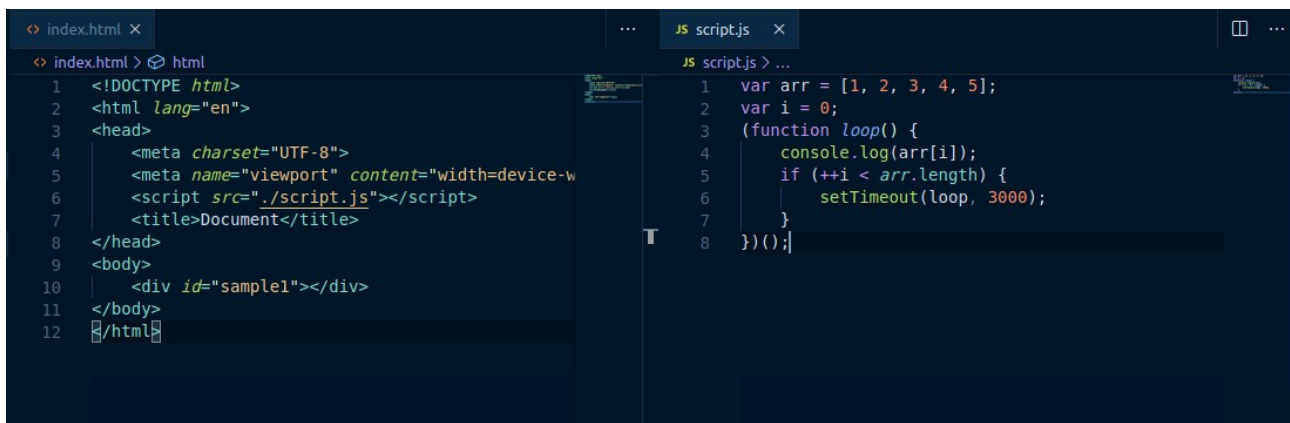
```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-w
6   <script src="./script.js"></script>
7   <title>Document</title>
8 </head>
9 <body>
10   <div id="sample1"></div>
11 </body>
12 </html>
```

The right pane shows `script.js` with the following content:

```
1 function person() {
2   this.isAdult = true;
3 }
4
5 function employee() {
6   this.isPaid = true;
7 }
8
9 employee.prototype = new person();
10
11 function developer() {
12   this.isTechie = true;
13 }
14
15 developer.prototype = new employee;
16
17 var sam = new developer();
18 console.log(sam.isAdult);
```



Q2. Given an array, say [1,2,3,4,5]. Print each element of an array after 3 secs.

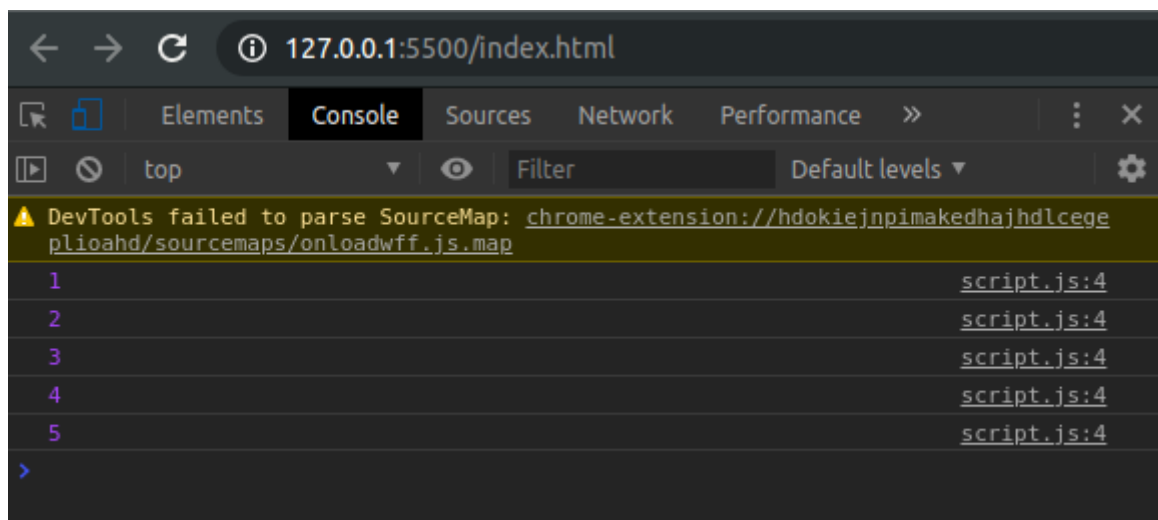


The screenshot shows a VS Code editor with two files open. The left pane shows `index.html` with the following content:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-w
6   <script src="./script.js"></script>
7   <title>Document</title>
8 </head>
9 <body>
10   <div id="sample1"></div>
11 </body>
12 </html>
```

The right pane shows `script.js` with the following content:

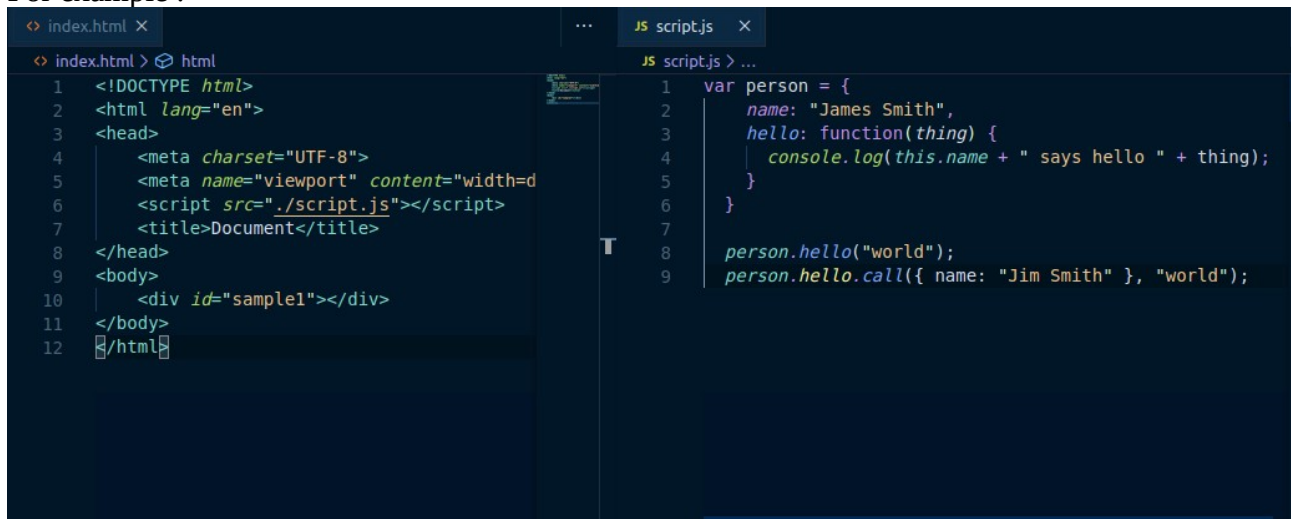
```
1 var arr = [1, 2, 3, 4, 5];
2 var i = 0;
3 (function loop() {
4   console.log(arr[i]);
5   if (++i < arr.length) {
6     setTimeout(loop, 3000);
7   }
8 })();
```



### Q3. Explain difference between Bind and Call (example).

Answer. Call attaches this into function and executes the function immediately

For example :-

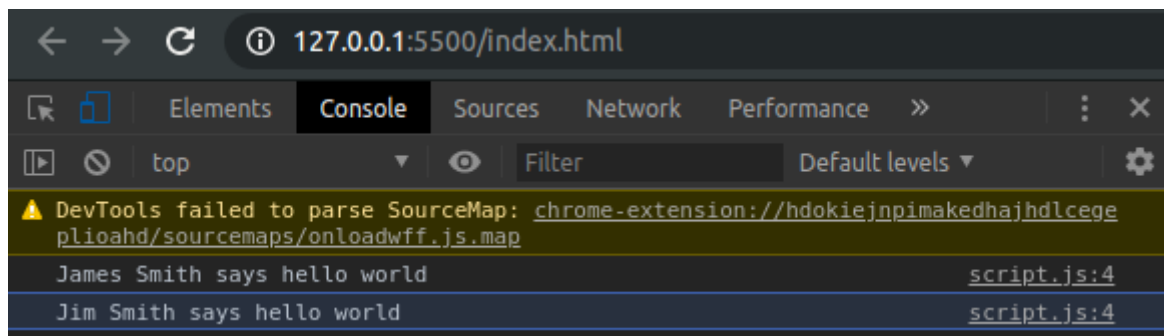


The screenshot shows two files in VS Code. The left pane shows `index.html` with the following content:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1">
6   <script src="./script.js"></script>
7   <title>Document</title>
8 </head>
9 <body>
10   <div id="sample1"></div>
11 </body>
12 </html>
```

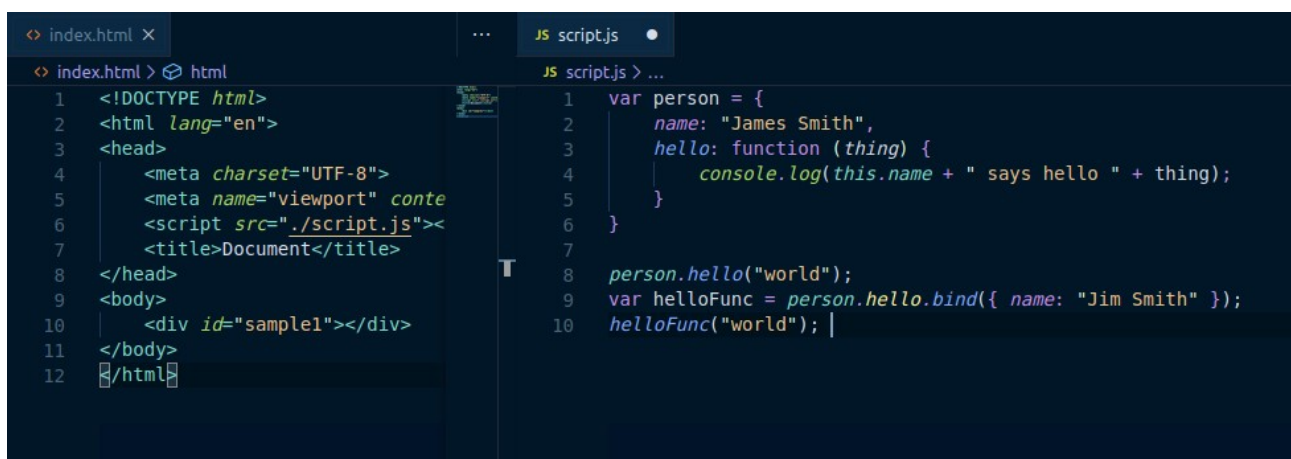
The right pane shows `JS script.js` with the following content:

```
1 var person = {
2   name: "James Smith",
3   hello: function(thing) {
4     console.log(this.name + " says hello " + thing);
5   }
6 }
7
8 person.hello("world");
9 person.hello.call({ name: "Jim Smith" }, "world");
```



Bind attaches this into function and it needs to be invoked separately.

For example :-

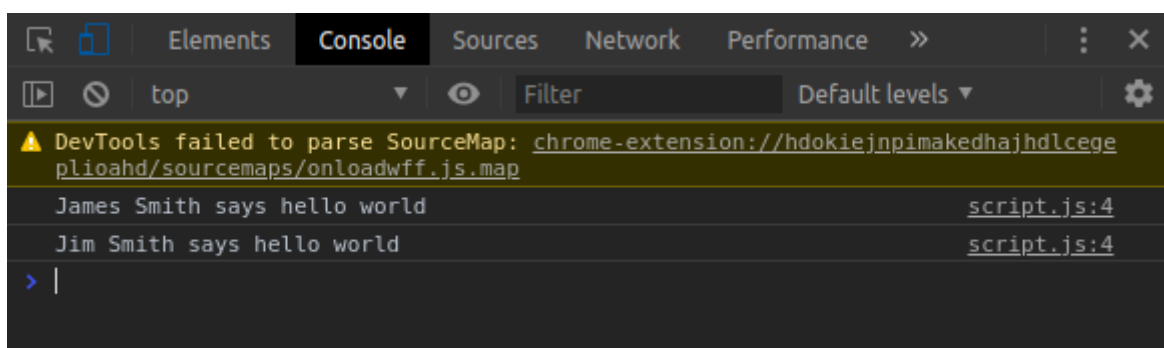


The screenshot shows two files in VS Code. The left pane shows `index.html` with the following content:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1">
6   <script src="./script.js"></script>
7   <title>Document</title>
8 </head>
9 <body>
10   <div id="sample1"></div>
11 </body>
12 </html>
```

The right pane shows `JS script.js` with the following content:

```
1 var person = {
2   name: "James Smith",
3   hello: function(thing) {
4     console.log(this.name + " says hello " + thing);
5   }
6 }
7
8 person.hello("world");
9 var helloFunc = person.hello.bind({ name: "Jim Smith" });
10 helloFunc("world");
```



#### Q4. Explain 3 properties of argument object.

A. Property of argument object :

##### 1. argument.callee

Callee is a property of the arguments object. It can be used to refer to the currently executing function inside the function body of that function. This is useful when the name of the function is unknown, such as within a function expression with no name (also called "anonymous functions").

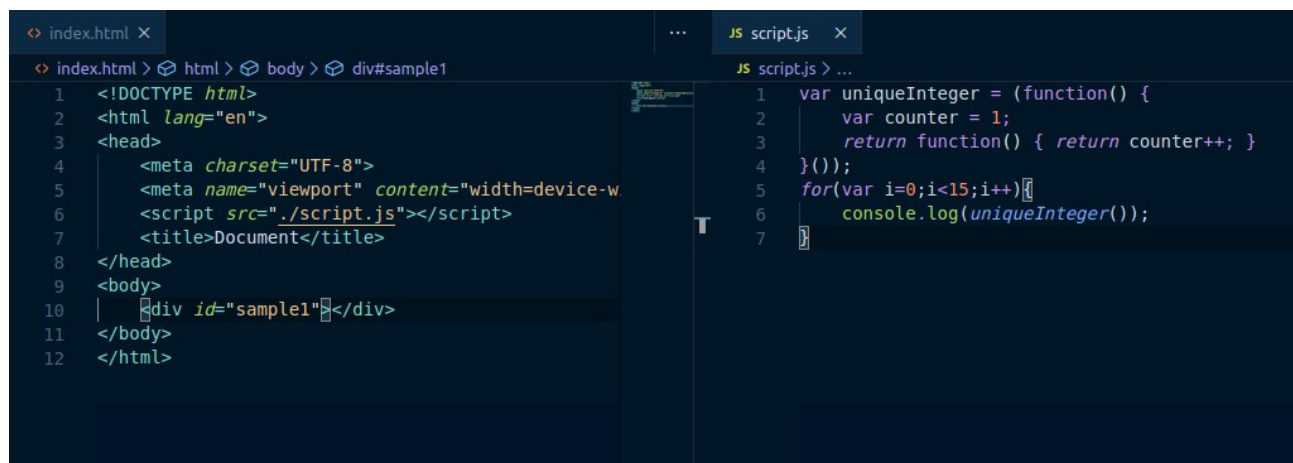
##### 2. argument.length

The arguments.length property provides the number of arguments actually passed to a function. This can be more or less than the defined parameter's count.

##### 3. arguments[@@iterator]()

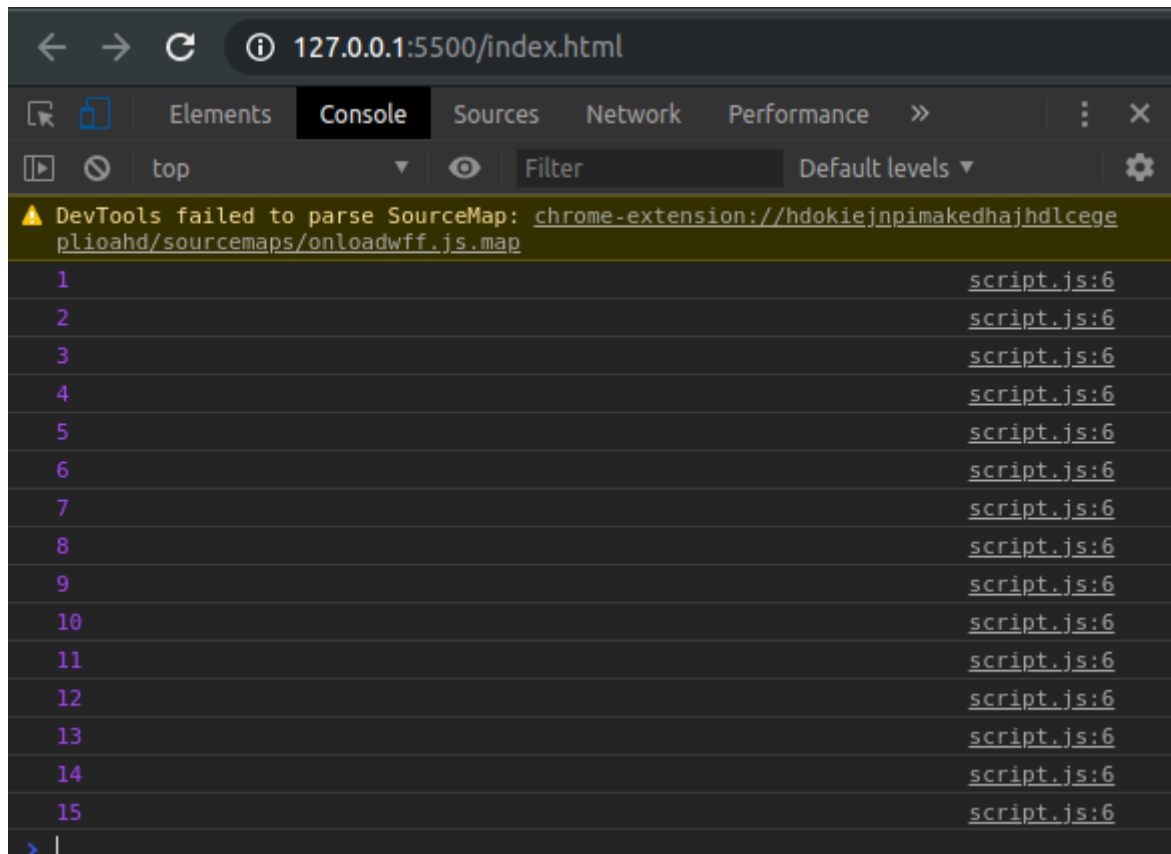
The initial value of the @@iterator property is the same function object as the initial value of the Array.prototype.values property.

#### Q6. Create a counter using closures.



```
index.html X
index.html > html > body > div#sample1
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-w
6   <script src="./script.js"></script>
7   <title>Document</title>
8 </head>
9 <body>
10   <div id="sample1"></div>
11 </body>
12 </html>

JS script.js X
JS script.js > ...
1 var uniqueInteger = (function() {
2   var counter = 1;
3   return function() { return counter++; }
4 })();
5 for(var i=0;i<15;i++){
6   console.log(uniqueInteger());
7 }
```

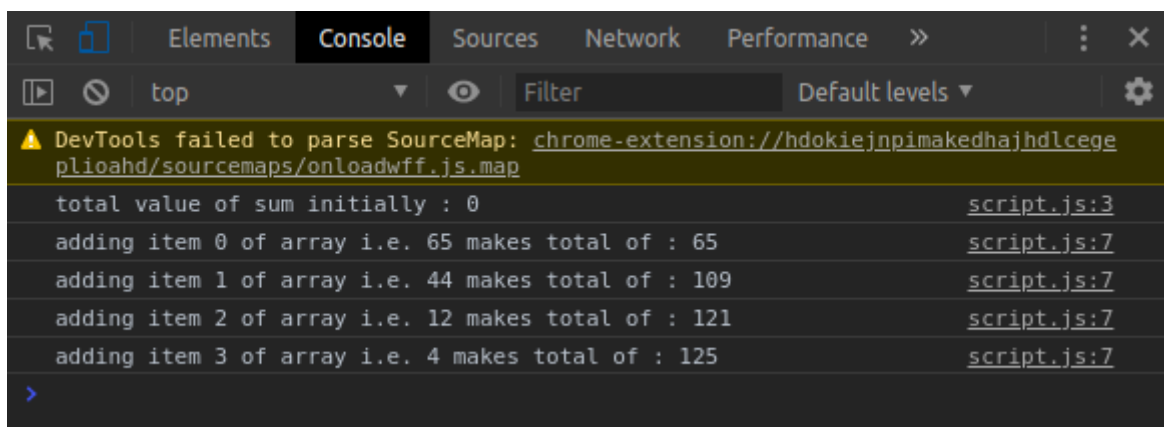
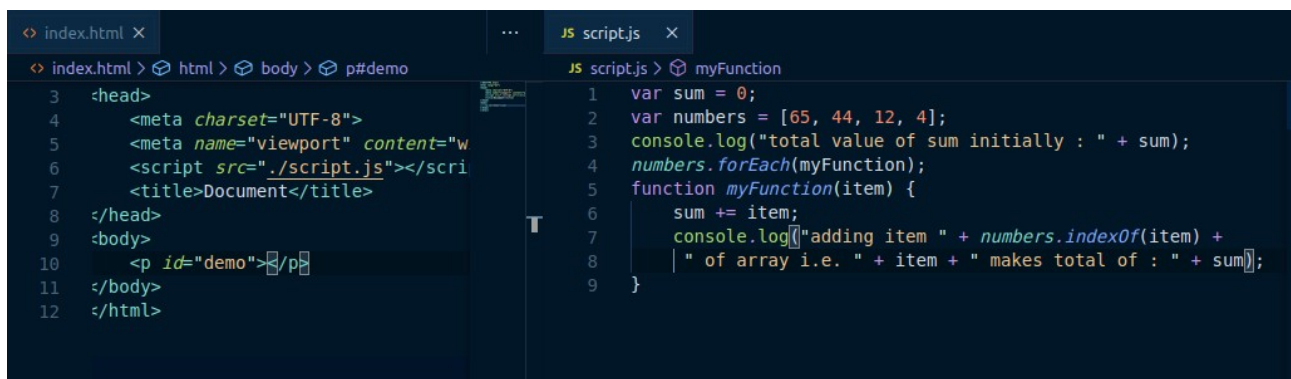


**Q7. Explain 5 array methods with example.**

Answer.

1. `forEach()`

The `forEach()` method calls a function once for each element in an array, in order.



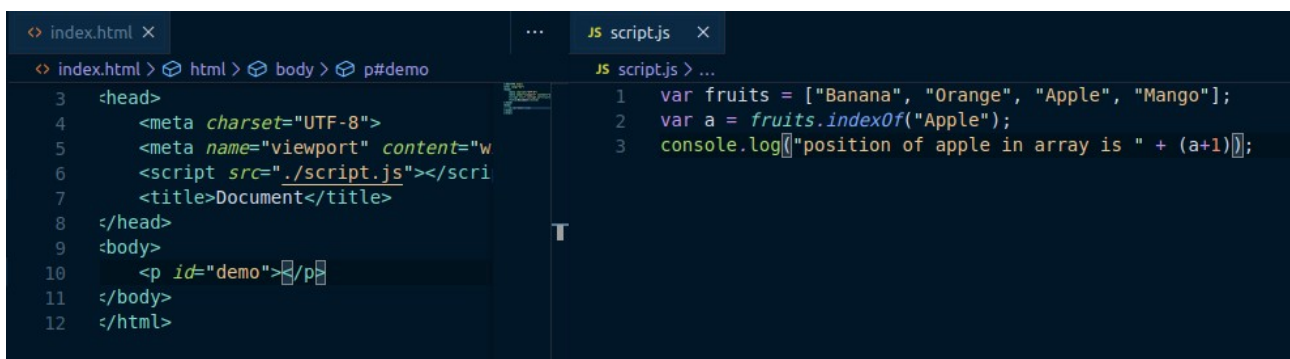
## 2. indexOf()

The `indexOf()` method searches the array for the specified item, and returns its position.

The search will start at the specified position, or at the beginning if no start position is specified, and end the search at the end of the array.

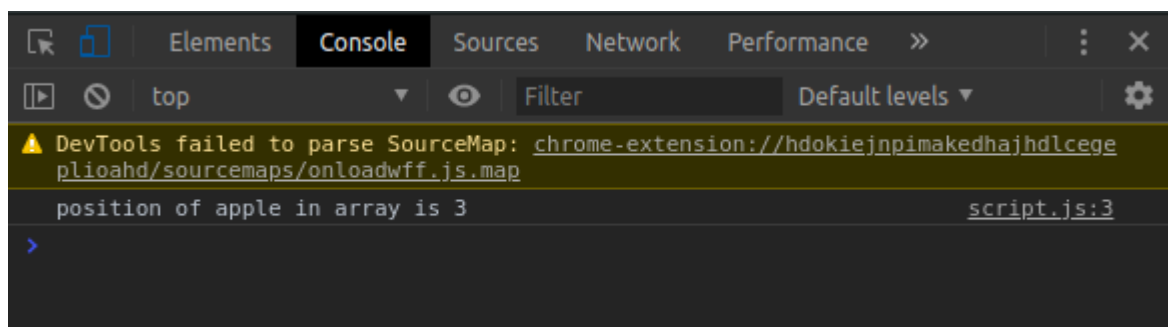
Returns -1 if the item is not found.

If the item is present more than once, the `indexOf` method returns the position of the first occurrence.



```
index.html > html > body > p#demo
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1">
6   <script src="./script.js"></script>
7   <title>Document</title>
8 </head>
9 <body>
10  <p id="demo"></p>
11 </body>
12 </html>

JS script.js > ...
1 var fruits = ["Banana", "Orange", "Apple", "Mango"];
2 var a = fruits.indexOf("Apple");
3 console.log("position of apple in array is " + (a+1));
```

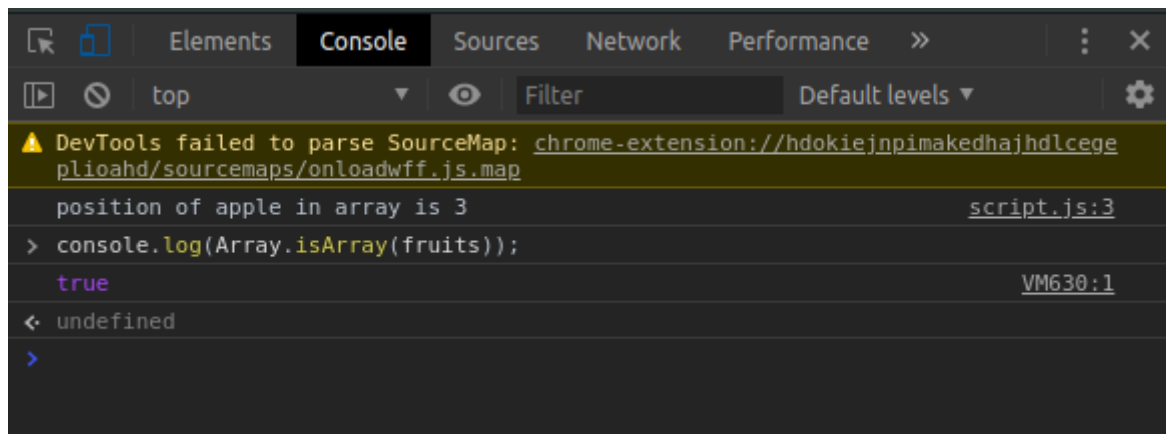


```
Elements Console Sources Network Performance >>
top Filter Default levels
⚠ DevTools failed to parse SourceMap: chrome-extension://hdokiejnpimakedhajhdlcegeplioahd/sourcemaps/onloadwff.js.map
position of apple in array is 3 script.js:3
>
```

## 3. isArray()

The `isArray()` method determines whether an object is an array.

This function returns true if the object is an array, and false if not.

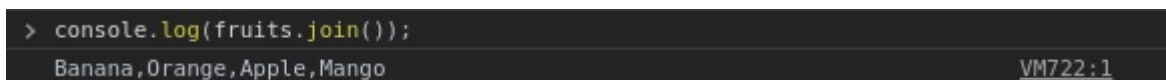


The screenshot shows the Chrome DevTools Console. At the top, there's a warning: "DevTools failed to parse SourceMap: chrome-extension://hdokiejnpimakedhajhdlcegeplioahd/sourcemaps/onloadwff.js.map". Below the warning, a log statement is visible: "position of apple in array is 3" with the source "script.js:3". The console also shows the execution of `console.log(Array.isArray(fruits));` which returns `true` (source: VM630:1), followed by an `undefined` value and a prompt character `>`.

#### 4. join()

The `join()` method returns the array as a string.

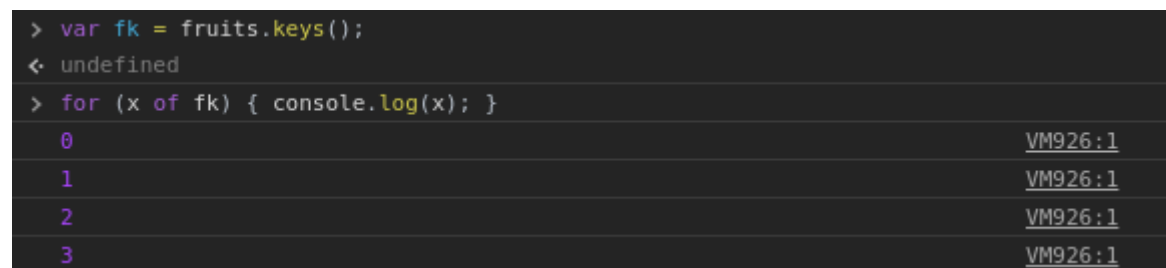
The elements will be separated by a specified separator. The default separator is comma (,).



The screenshot shows a single line of code in the console: `console.log(fruits.join());`. The output is the string "Banana,Orange,Apple,Mango" (source: VM722:1).

#### 5. keys()

The `keys()` method returns an Array Iterator object with the keys of an array.



The screenshot shows a sequence of code and output in the console. First, `var fk = fruits.keys();` is executed, returning `undefined`. Then, a `for` loop is used to iterate over the keys: `for (x of fk) { console.log(x); }`. The output shows the indices 0, 1, 2, and 3, each on a new line (all sources are VM926:1).