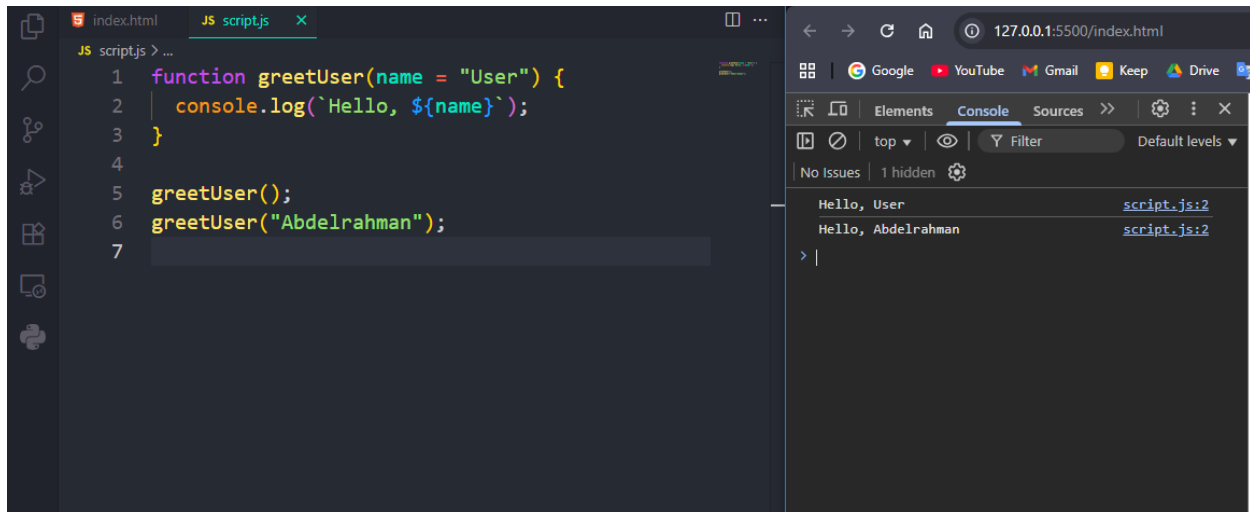


JavaScript ES6 – Day.7 Assignments

1- Write a function that greets a user, using a default parameter for the name.



The screenshot shows a web browser with a JavaScript file named `script.js` open in the editor. The code defines a function `greetUser` with a default parameter `name = "User"`. The function logs a greeting message to the console. The browser's console shows the output of the function calls.

```
1 function greetUser(name = "User") {  
2   console.log(`Hello, ${name}`);  
3 }  
4  
5 greetUser();  
6 greetUser("Abdelrahman");  
7
```

The console output shows:

- Hello, User (from `script.js:2`)
- Hello, Abdelrahman (from `script.js:2`)

2- Write a function that calculates the total price with a default tax rate parameter.



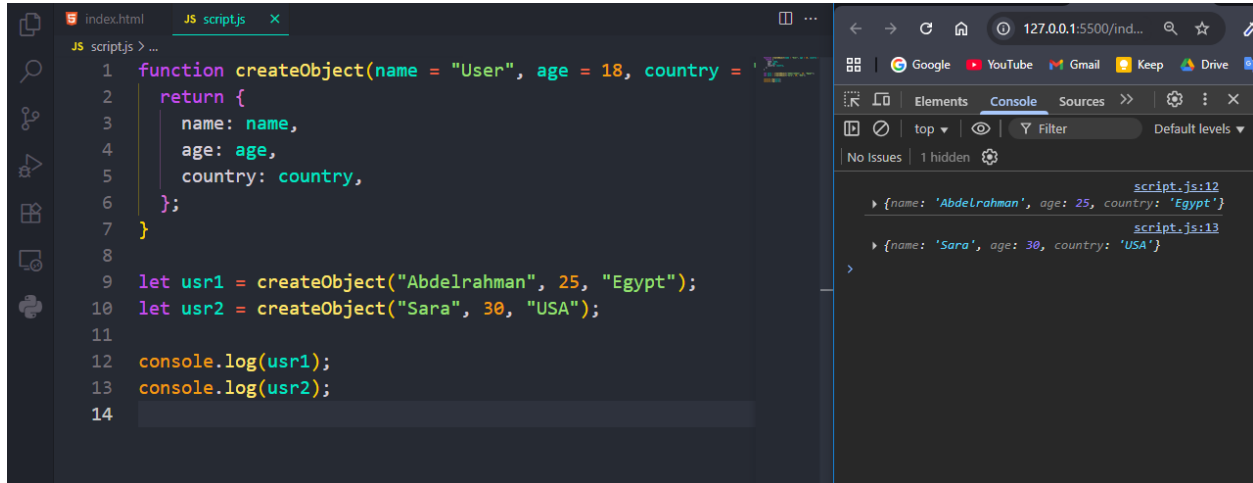
The screenshot shows a web browser with a JavaScript file named `script.js` open in the editor. The code defines a function `totalPrice` with a default parameter `tax = 0.1`. The function returns the total price calculated by adding the price and the tax. The browser's console shows the output of the function calls.

```
1 function totalPrice(price, tax = 0.1) {  
2   return price + price * tax;  
3 }  
4  
5 console.log(totalPrice(100));  
6 console.log(totalPrice(100, 0.15));  
7
```

The console output shows:

- 110 (from `script.js:5`)
- 115 (from `script.js:6`)

3- Write a function that creates a user object, using a default role parameter.



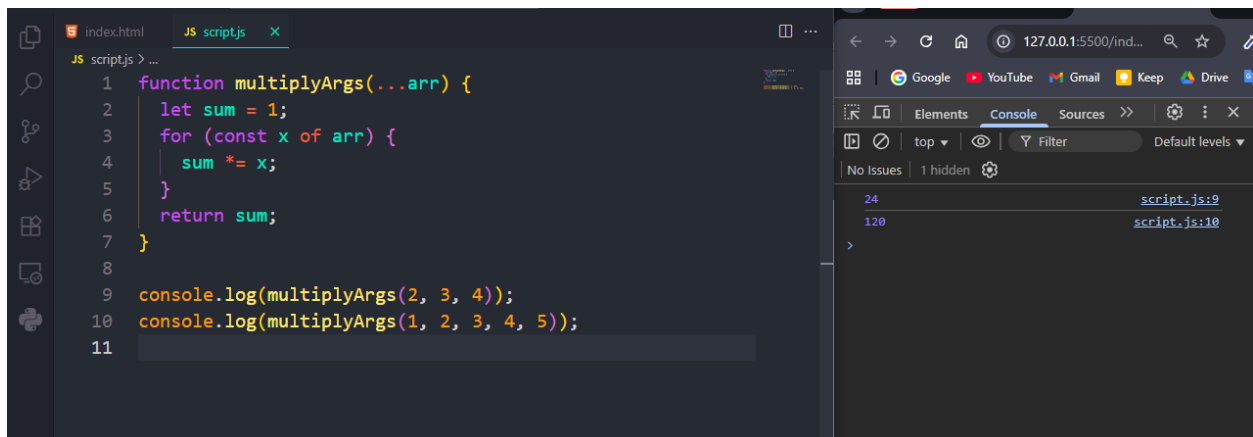
The screenshot shows a web browser with a JavaScript file named 'script.js' loaded. The function 'createObject' is defined with default parameters: 'name' is 'User', 'age' is 18, and 'country' is 'Egypt'. It returns an object with these properties. Two user objects are created: 'usr1' (Abdelrahman, 25, Egypt) and 'usr2' (Sara, 30, USA). The console shows the log output for both objects.

```
1 function createObject(name = "User", age = 18, country = "Egypt") {
2   return {
3     name: name,
4     age: age,
5     country: country,
6   };
7 }
8
9 let usr1 = createObject("Abdelrahman", 25, "Egypt");
10 let usr2 = createObject("Sara", 30, "USA");
11
12 console.log(usr1);
13 console.log(usr2);
14
```

Console output:

```
{name: 'Abdelrahman', age: 25, country: 'Egypt'}
{name: 'Sara', age: 30, country: 'USA'}
```

4- Write a function that multiplies any number of arguments using the rest operator.



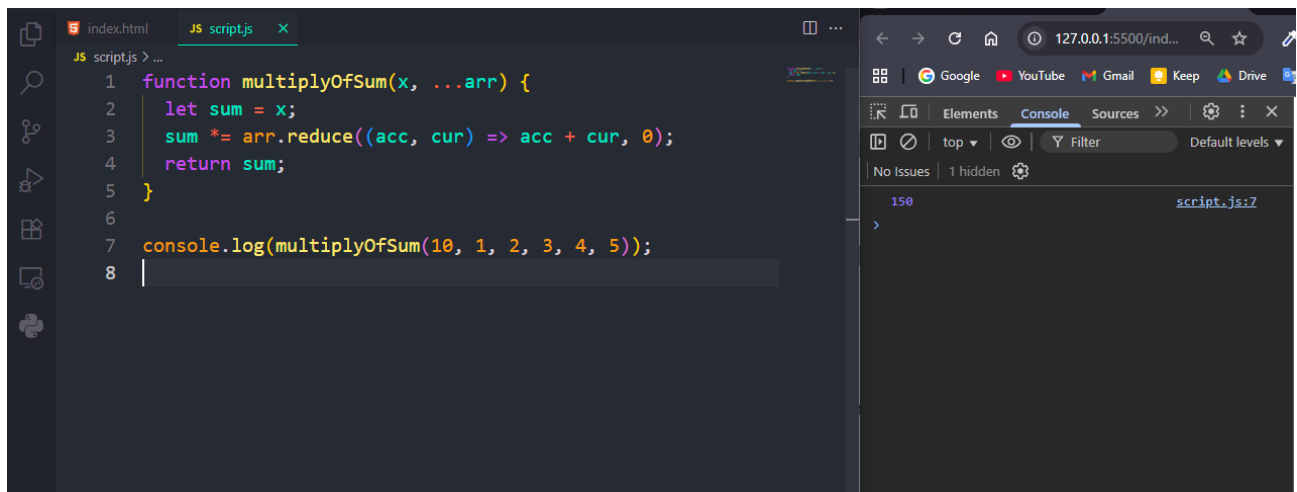
The screenshot shows a web browser with a JavaScript file named 'script.js' loaded. The function 'multiplyArgs' is defined using the rest operator '...arr'. It calculates the product of all arguments. The console shows the log output for two calls: 'multiplyArgs(2, 3, 4)' which returns 24, and 'multiplyArgs(1, 2, 3, 4, 5)' which returns 120.

```
1 function multiplyArgs(...arr) {
2   let sum = 1;
3   for (const x of arr) {
4     sum *= x;
5   }
6   return sum;
7 }
8
9 console.log(multiplyArgs(2, 3, 4));
10 console.log(multiplyArgs(1, 2, 3, 4, 5));
11
```

Console output:

```
24
120
```

5- Write a function that multiplies the first argument by the sum of all the rest using the rest operator.



The screenshot shows a web browser with a JavaScript file named `script.js` open. The code defines a function `multiplyOfSum(x, ...arr)` that calculates the product of the first argument `x` and the sum of the rest of the arguments `arr`. The function uses `arr.reduce((acc, cur) => acc + cur, 0)` to sum the array elements. The function is called with `multiplyOfSum(10, 1, 2, 3, 4, 5)`, and the result `150` is logged to the console.

```
1 function multiplyOfSum(x, ...arr) {  
2   let sum = x;  
3   sum *= arr.reduce((acc, cur) => acc + cur, 0);  
4   return sum;  
5 }  
6  
7 console.log(multiplyOfSum(10, 1, 2, 3, 4, 5));  
8
```

The console output shows the value `150` at line 7 of `script.js`.

6- Write a function that takes a variable number of strings and returns them as a single array using the rest operator.

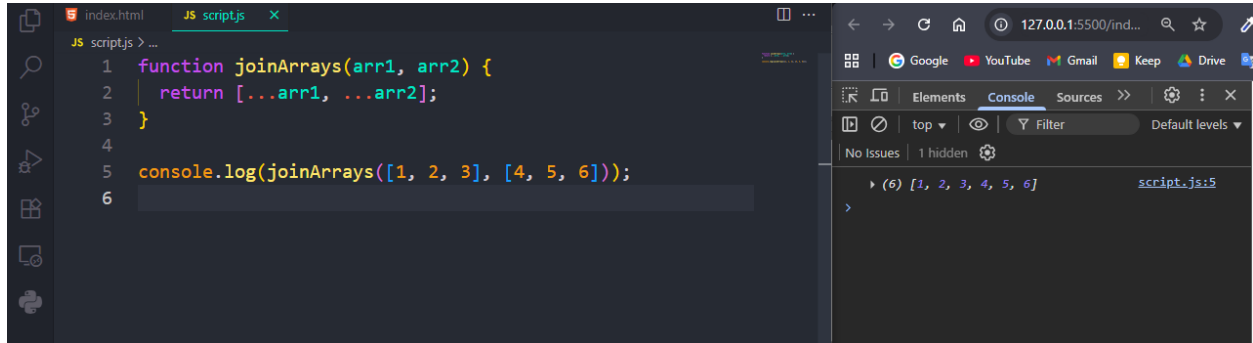


The screenshot shows a web browser with a JavaScript file named `script.js` open. The code defines a function `joinStrings(...strs)` that returns the rest of the arguments `strs` as an array. The function is called with `joinStrings("Hello", "world", "!")`, and the result is logged to the console.

```
1 function joinStrings(...strs) {  
2   return strs;  
3 }  
4  
5 console.log(joinStrings("Hello", "world", "!"));  
6
```

The console output shows the array `(3) ['Hello', 'world', '!']` at line 5 of `script.js`.

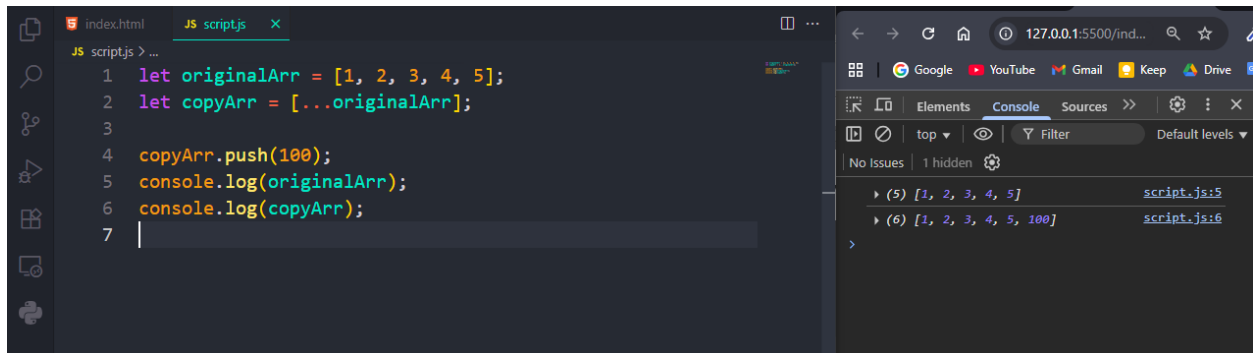
7- Create a new array by combining two arrays using the spread operator.



The screenshot shows a web browser with a JavaScript file named `script.js` open. The code defines a function `joinArrays(arr1, arr2)` that returns a new array created by spreading `arr1` and `arr2`. The function is called with `[1, 2, 3]` and `[4, 5, 6]`, and the result is logged to the console. The console output shows the array `[1, 2, 3, 4, 5, 6]`.

```
1 function joinArrays(arr1, arr2) {  
2   return [...arr1, ...arr2];  
3 }  
4  
5 console.log(joinArrays([1, 2, 3], [4, 5, 6]));  
6
```

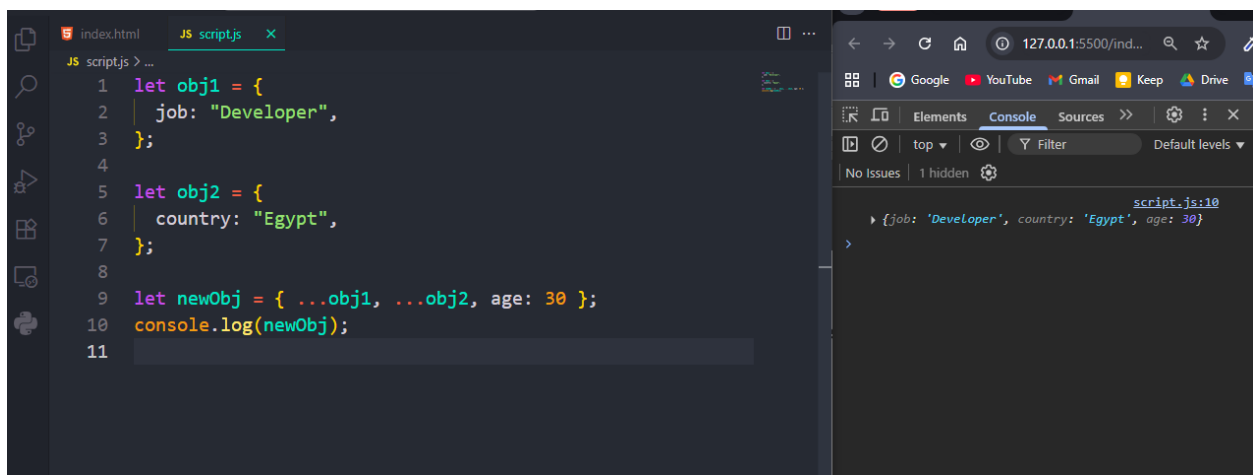
8- Copy an array using the spread operator.



The screenshot shows a web browser with a JavaScript file named `script.js` open. The code creates an array `originalArr` with values `[1, 2, 3, 4, 5]`, copies it into `copyArr` using the spread operator, and then pushes the value `100` into `copyArr`. The original array and the modified copy are both logged to the console. The console output shows `[1, 2, 3, 4, 5]` for the original array and `[1, 2, 3, 4, 5, 100]` for the copy.

```
1 let originalArr = [1, 2, 3, 4, 5];  
2 let copyArr = [...originalArr];  
3  
4 copyArr.push(100);  
5 console.log(originalArr);  
6 console.log(copyArr);  
7
```

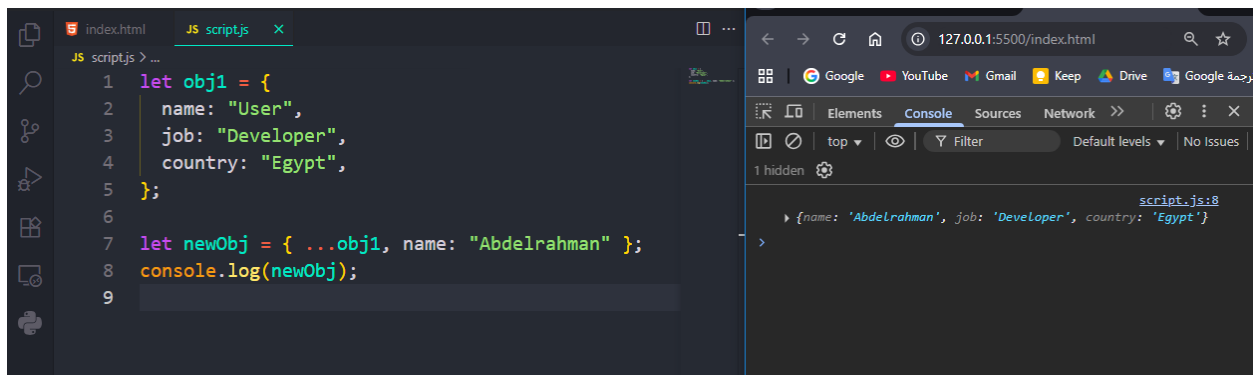
9- Merge two objects into one using the spread operator.



The screenshot shows a web browser with a JavaScript file named `script.js` open. The code creates two objects: `obj1` with `job: "Developer"` and `obj2` with `country: "Egypt"`. A new object `newObj` is created by spreading `obj1` and `obj2` and adding `age: 30`. The new object is logged to the console. The console output shows the merged object: `{job: 'Developer', country: 'Egypt', age: 30}`.

```
1 let obj1 = {  
2   job: "Developer",  
3 };  
4  
5 let obj2 = {  
6   country: "Egypt",  
7 };  
8  
9 let newObj = { ...obj1, ...obj2, age: 30 };  
10 console.log(newObj);  
11
```

10- Update a property in an object using the spread operator to create a new object.

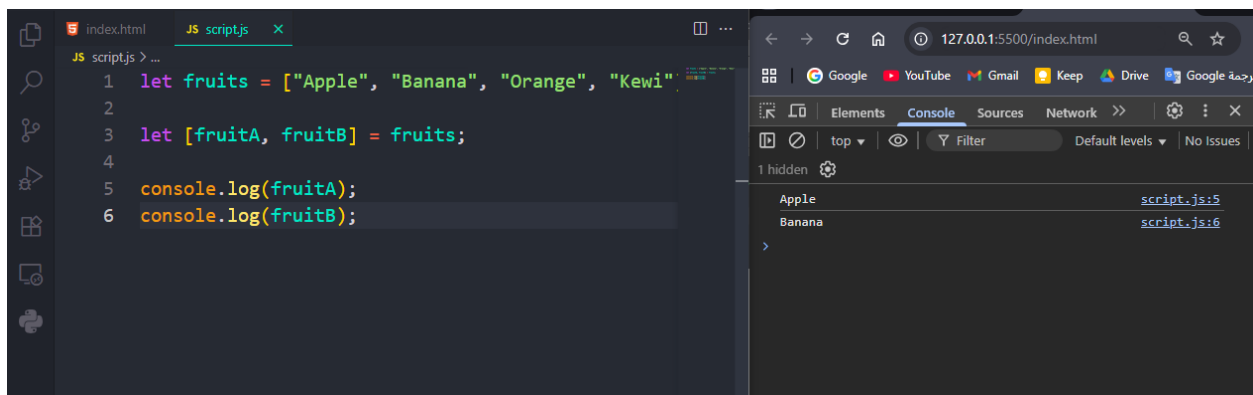


The screenshot shows a code editor with the following JavaScript code in `script.js`:

```
1 let obj1 = {  
2   name: "User",  
3   job: "Developer",  
4   country: "Egypt",  
5 };  
6  
7 let newObj = { ...obj1, name: "Abdelrahman" };  
8 console.log(newObj);  
9
```

The browser console on the right shows the output of `console.log(newObj)` as an object: `{name: 'Abdelrahman', job: 'DeveLoper', country: 'Egypt'}`.

11- Destructure an array to get the first and second elements into variables.

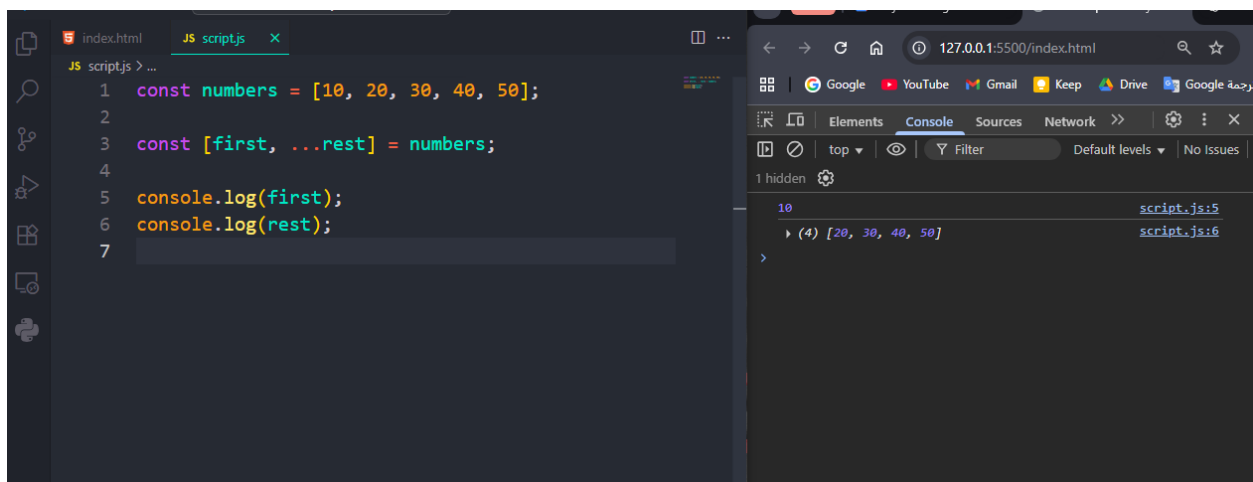


The screenshot shows a code editor with the following JavaScript code in `script.js`:

```
1 let fruits = ["Apple", "Banana", "Orange", "Kewi"];  
2  
3 let [fruitA, fruitB] = fruits;  
4  
5 console.log(fruitA);  
6 console.log(fruitB);
```

The browser console on the right shows the output of `console.log(fruitA)` as `Apple` and `console.log(fruitB)` as `Banana`.

12- Destructure an array to get the first element and the rest into another array.

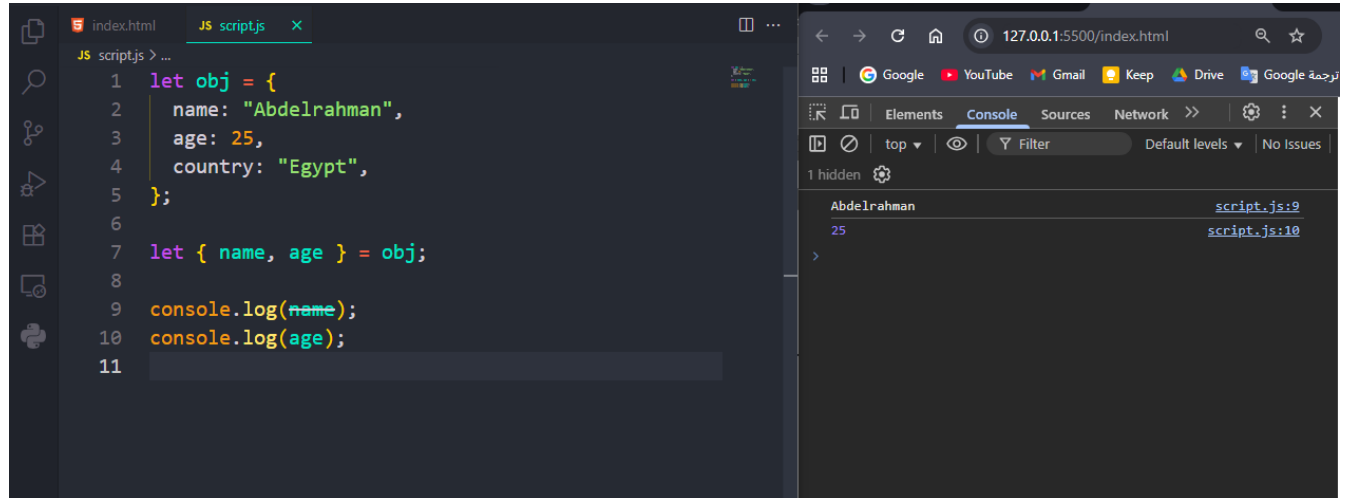


The screenshot shows a code editor with the following JavaScript code in `script.js`:

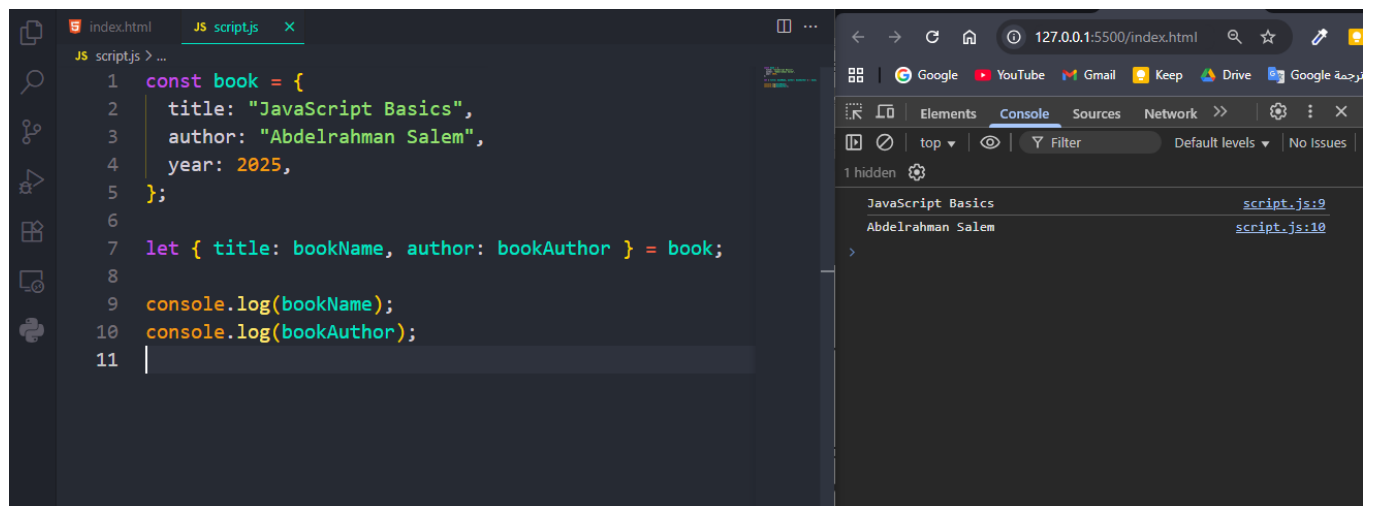
```
1 const numbers = [10, 20, 30, 40, 50];  
2  
3 const [first, ...rest] = numbers;  
4  
5 console.log(first);  
6 console.log(rest);  
7
```

The browser console on the right shows the output of `console.log(first)` as `10` and `console.log(rest)` as an array: `(4) [20, 30, 40, 50]`.

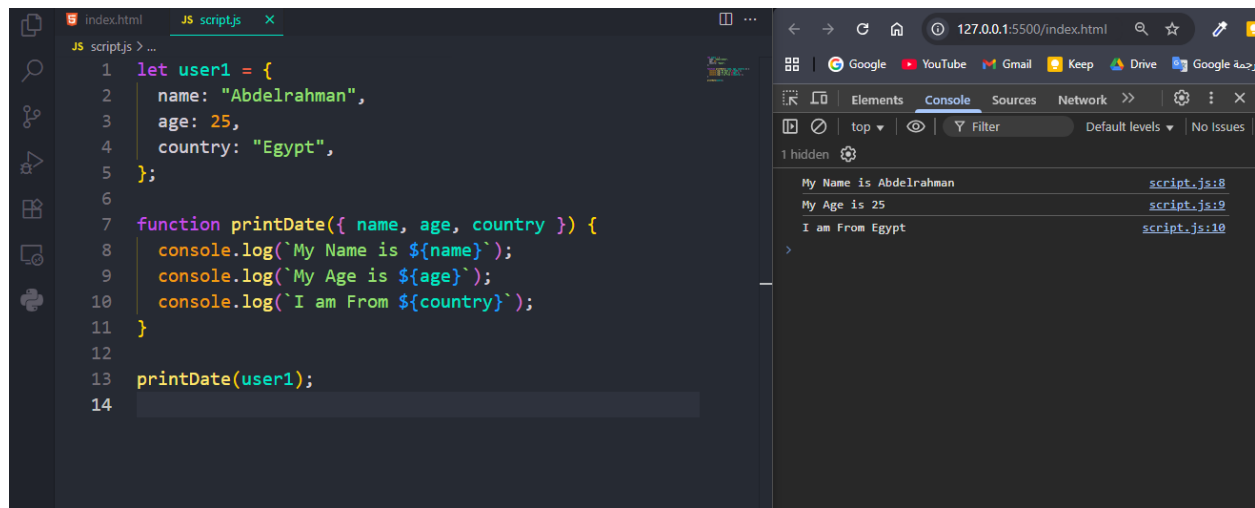
13- Destructure an object to extract two properties into variables.



14- Destructure an object and rename the extracted properties.



15- Write a function that takes an object as a parameter and uses destructuring in the parameter list to extract specific properties.



The screenshot displays a web browser with a local address bar showing `127.0.0.1:5500/index.html`. The browser's developer console is open, showing three log messages: "My Name is Abdelrahman" (script.js:8), "My Age is 25" (script.js:9), and "I am From Egypt" (script.js:10). The code in the background defines a `user1` object and a `printDate` function that uses destructuring to access its properties.

```
1 let user1 = {
2   name: "Abdelrahman",
3   age: 25,
4   country: "Egypt",
5 };
6
7 function printDate({ name, age, country }) {
8   console.log(`My Name is ${name}`);
9   console.log(`My Age is ${age}`);
10  console.log(`I am From ${country}`);
11 }
12
13 printDate(user1);
14
```