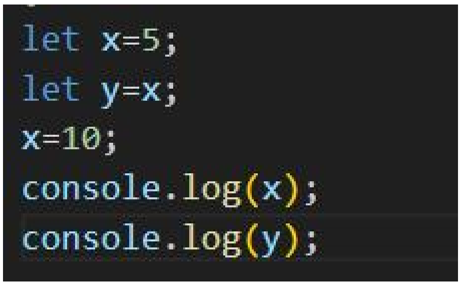
1. what will be the output?

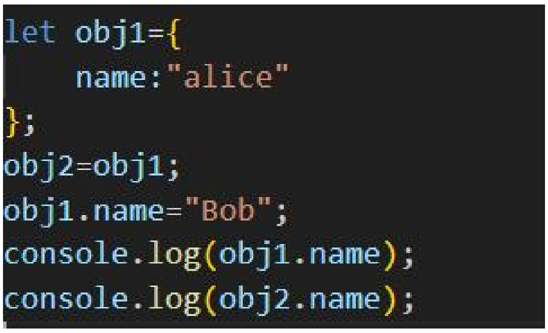


```
let x=5;
let y=x;
x=10;
console.log(x);
console.log(y);
```

o/p: 10,5

Explanation: At first 5 is assigned to x and assigned x (x=5) value to y. Here x and y variables hold same value (5). x is reassigned to 10 but y is still 5 because it was assigned before the change to x. Then the output will be 10 and 5.

2.what will be the output?

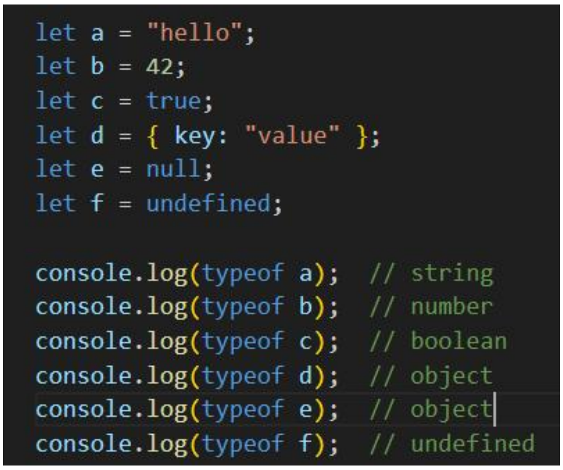


```
let obj1={
    name:"alice"
};
obj2=obj1;
obj1.name="Bob";
console.log(obj1.name);
console.log(obj2.name);
```

o/p: Bob, Bob

Explanation: obj1 is an object. obj2 = obj1; creates a reference to the same object, meaning both obj1 and obj2 refer to the same memory location. Changing obj1.name = "Bob"; changes the name property of the object that both obj1 and obj2 reference. Then the output will be Bob and Bob.

3.what will be the output?



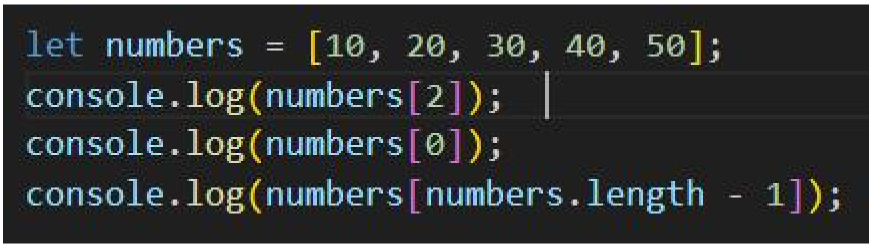
```
let a = "hello";
let b = 42;
let c = true;
let d = { key: "value" };
let e = null;
let f = undefined;

console.log(typeof a);  // string
console.log(typeof b);  // number
console.log(typeof c);  // boolean
console.log(typeof d);  // object
console.log(typeof e);  // object
console.log(typeof f);  // undefined
```

Explanation:

- **typeof a** outputs "string" because a is a string.
- **typeof b** outputs "number" because b is a number.
- **typeof c** outputs "boolean" because c is a boolean.
- **typeof d** outputs "object" because objects in JavaScript have "object" type.
- **typeof e** outputs "object".
- **typeof f** outputs "undefined" because f is declared as undefined.

4.what will be the output?



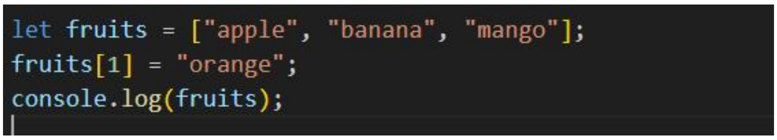
```
let numbers = [10, 20, 30, 40, 50];
console.log(numbers[2]);
console.log(numbers[0]);
console.log(numbers[numbers.length - 1]);
```

o/p: 30,10,50

Explanation:

- numbers[2] accesses the third element in the array, which is 30.
- numbers[0] accesses the first element, which is 10.
- numbers[numbers.length - 1] accesses the last element, which is 50, because of negative indexing.

5. what will be the output?



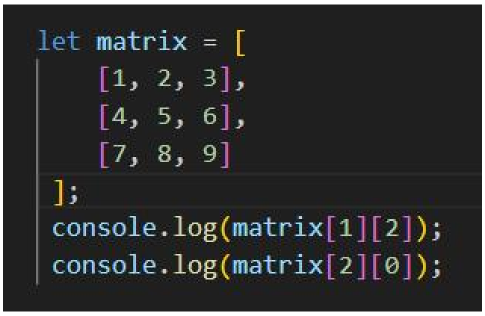
```
let fruits = ["apple", "banana", "mango"];
fruits[1] = "orange";
console.log(fruits);
```

o/p: ['apple', 'orange', 'mango']

Explanation:

- The original array contains ["apple", "banana", "mango"].
- fruits[1] = "orange"; replaces the second element ("banana") with "orange".

6. what will be the output?



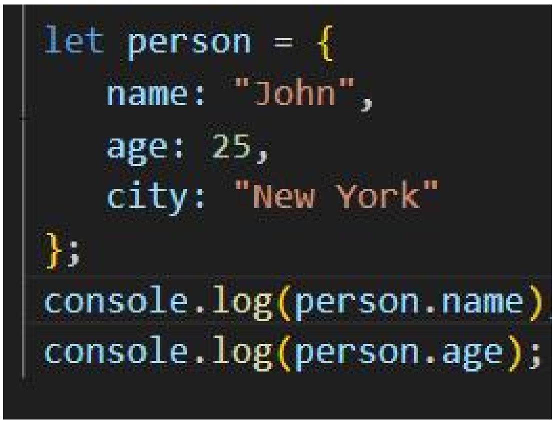
```
let matrix = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
 ];
 console.log(matrix[1][2]);
 console.log(matrix[2][0]);
```

o/p: 6,7

Explanation:

- matrix[1][2] accesses the element in the second row, third column of the matrix, which is 6.
- matrix[2][0] accesses the element in the third row, first column of the matrix, which is 7.

7. what will be the output?



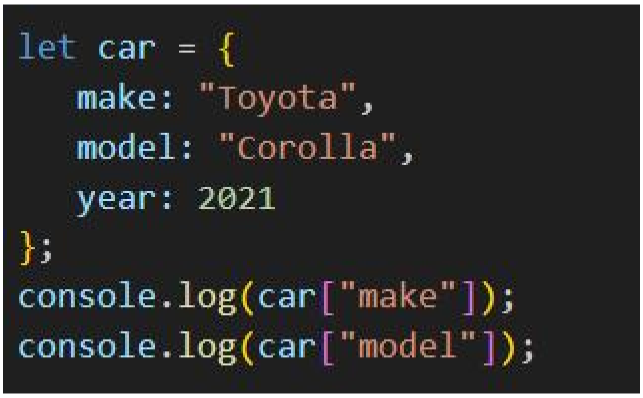
```
let person = {
   name: "John",
   age: 25,
   city: "New York"
};
console.log(person.name)
console.log(person.age);
```

o/p: John,25

Explanation:

- person.name accesses the name property of the person object, which is "John".
- person.age accesses the age property, which is 25.

8. what will be the output?



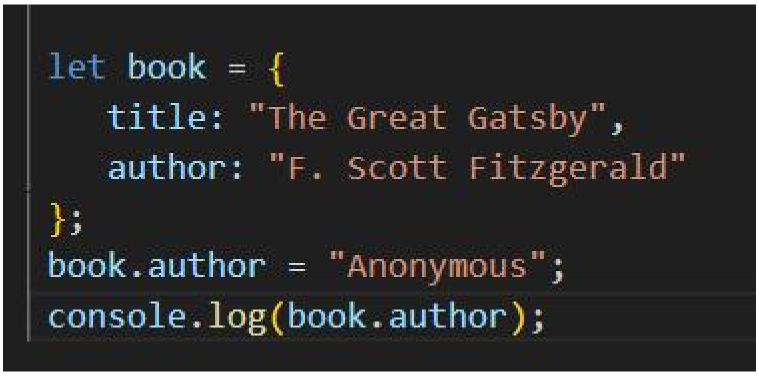
```
let car = {
   make: "Toyota",
   model: "Corolla",
   year: 2021
};
console.log(car["make"]);
console.log(car["model"]);
```

o/p: Toyota, Corolla

Explanation:

- car["make"] accesses the make property of the car object, which is "Toyota".
- car["model"] accesses the model property, which is "Corolla".

9. what will be the output?



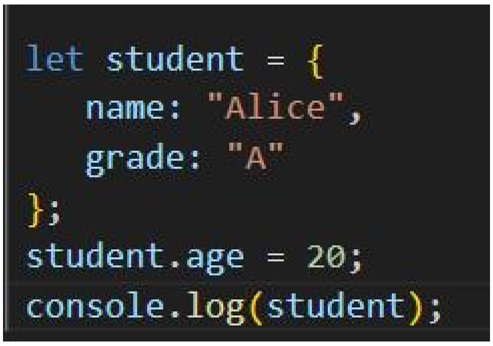
```
let book = {
   title: "The Great Gatsby",
   author: "F. Scott Fitzgerald"
};
book.author = "Anonymous";
console.log(book.author);
```

o/p: Anonymous

Explanation:

- The book.author property value is "F. Scott Fitzgerald".
- book.author = "Anonymous"; changes the author property to "Anonymous".
- console.log(book.author); outputs "Anonymous" here the author property was updated.

10. what will be the output?



```
let student = {
   name: "Alice",
   grade: "A"
};
student.age = 20;
console.log(student);
```

o/p: {name: 'Alice', grade: 'A', age: 20}

Explanation:

- The student object initially contains name and grade properties.
- Here 20 is assigned to the age property to the object.
- Outputs the complete object with the new age property.