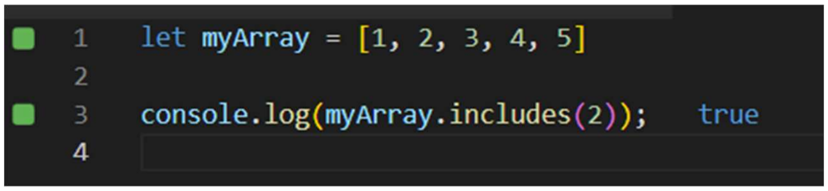
1. Select **five methods** that can be used on an Array and describe the following for each:

1) what the method signature is

2) what the method does

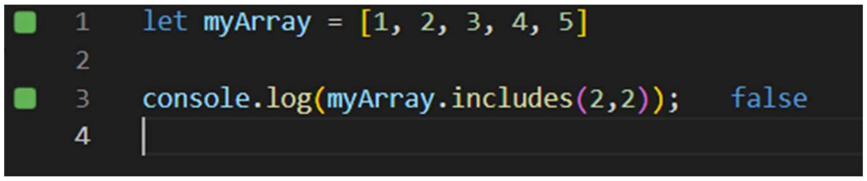
3) why would this method be useful (how could you use it)?

Array.includes() is a method in JavaScript for checking an array for a specific value. It returns a Boolean value; true if the element is in the searched array, and false if it is not. You can also choose the starting point for the search. We can see this in action with the following example:



```
let myArray = [1, 2, 3, 4, 5]

console.log(myArray.includes(2));   true
```

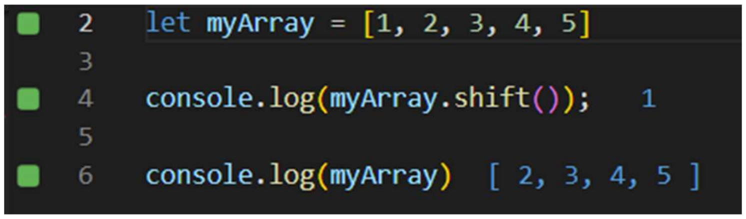
In the above example, we printed myArray.includes(2); which searches the array myArray for the number 2. 2 is in the array, so it returns true. Let's try starting from a specific point in the array-



```
let myArray = [1, 2, 3, 4, 5]

console.log(myArray.includes(2,2));   false
```

In this example, we are still searching myArray for the number 2, but we are starting at index 2, which starts at the number 3 in the array. Since there is no 2 in the array *when starting from this point*, false is returned. If you do not specify where to start the search, .includes() will search the entire array by default. This method is very useful for cases where you have an array with hundreds of elements and you need to check for a specific value within the array. Instead of manually searching through the array(if possible), you can use .includes() to find out in a snap if the desired value is in the array or not.

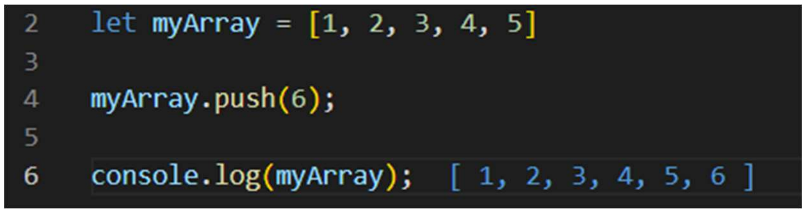
Array.shift() is a method in JavaScript for removing and returning the first element of an array. Let's look at it in action:



```
let myArray = [1, 2, 3, 4, 5]

console.log(myArray.shift());   1

console.log(myArray)  [ 2, 3, 4, 5 ]
```

In the above example, we see that printing myArray.shift() returns the value 1, and if we print the array again, it no longer contains the element 1 that was at index[0]. This method would be useful for a game where you need to draw the top(first) card off a deck of cards and have the next card (previously at index[1] become the top card.

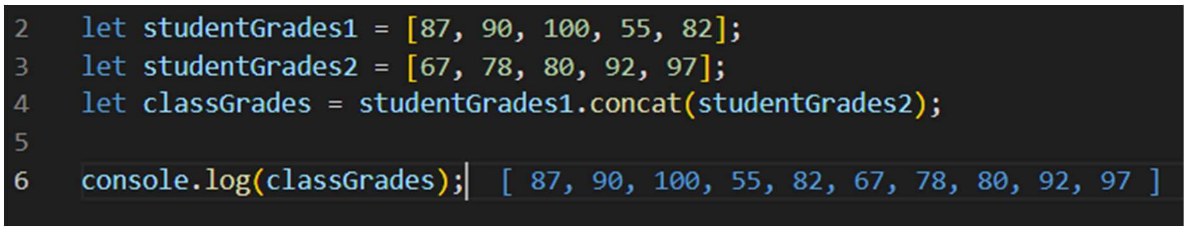
Array.push() is a method of JavaScript used for adding elements to an array- specifically to the end of an array.



```
let myArray = [1, 2, 3, 4, 5]

myArray.push(6);

console.log(myArray);  [ 1, 2, 3, 4, 5, 6 ]
```

In the example above, we have myArray but we need to add the another number to that array. We can use the .push() method to quickly and easily add elements to an array. A good use for this would be adding grades in a class to then calculate the average, or names on a list of people coming to a party.

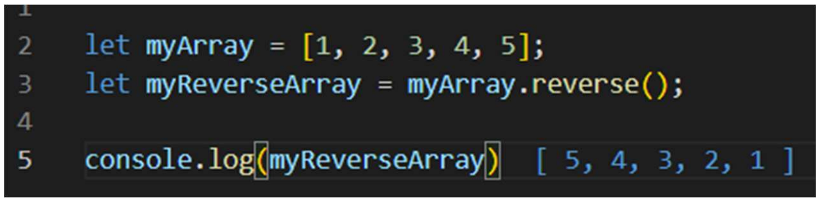
Array.concat() is a method of JavaScript used for joining multiple arrays into a new array. In the below example, we have two arrays of student grades, but we need to combine them into a new array (classGrades) to see the all the grades in one new array. The method signature for array.concat() is *array1.concat(array2);*



```
let studentGrades1 = [87, 90, 100, 55, 82];
let studentGrades2 = [67, 78, 80, 92, 97];
let classGrades = studentGrades1.concat(studentGrades2);

console.log(classGrades);  [ 87, 90, 100, 55, 82, 67, 78, 80, 92, 97 ]
```

You can also combine more than two arrays in one go using array.concat() by adding them with commas, such as, *array1.concat(array2, array3, array4);* . This will combine 4 total arrays into one new array.

Array.reverse() is a method of JavaScript used for reversing the elements of an array, so the last is first, and vice versa. This is useful for changing how data is displayed, such as arranging comments or messages from oldest to newest. Let’s see it in action below-



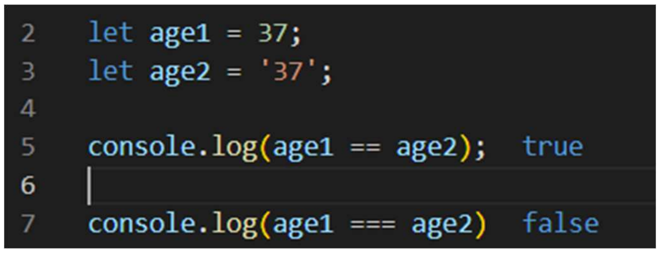
```
let myArray = [1, 2, 3, 4, 5];
let myReverseArray = myArray.reverse();

console.log(myReverseArray)  [ 5, 4, 3, 2, 1 ]
```

In this example, we used the .reverse() method to create a new array containing the elements from myArray, only now in reversed order.

2. What is the difference between == and ===?

Strict equality (===) is a Boolean operator that compares two values for equality of **both value and data type.** If both values are strictly equal, the result will return *true*. Otherwise, it will return *false*.

Loose equality (==) is a Boolean operator that compares two values for equality of value. Using == will perform a type conversion when comparing values.



```
let age1 = 37;
let age2 = '37';

console.log(age1 == age2);   true

console.log(age1 === age2)   false
```

In the above example, we can see two variables *age1* and *age2*. When compared with ==, it will convert them to the same data type before comparing 37 and 37, returning *true*. When using ===, since they do not share the same data type, they are not *strictly equal* and therefore we see that it returns *false*.

https://www.w3schools.com/js/js_array_methods.asp

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Equality_comparisons_and_sameness