Practice Questions on Arrays, While Loops and Boolean Logic

3 questions

Submit Quiz

1.

A student realises he needs to get more organised. He develops JavaScript code to help him. Tasks he has to do are stored in the array *todo* and the priority of those items are stored in array*priority*. In that array the number 1 means the highest priority, 2 means the second highest priority, and so on. His code will then display the items he needs to do, in order from the highest priority item to the lowest priority item.

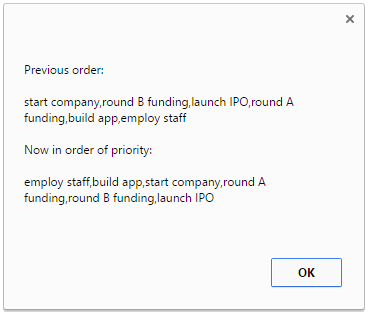
For example, if the data is stored like this:

var todo=["start company", "round B funding", "launch IPO",

"round A funding", "build app", "employ staff"];

var priority=[3, 5, 6, 4, 2, 1];

then the code needs to show this result:



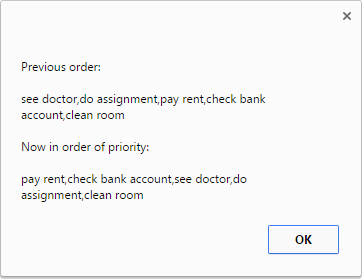
Here is another example. If the data is stored like this:

var todo = ["see doctor", "do assignment", "pay rent",

"check bank account", "clean room"];

var priority = [3, 4, 1, 2, 5];

then the code needs to shows this result:



Here is the code written by the student.

<!doctype html>

<html>

<head>

<script>

var todo = ["see doctor", "do assignment", "pay rent",

"check bank account", "clean room"];

var priority = [3, 4, 1, 2, 5];

var index = 0;

var todoInOrderOfPriority = [];

while ( index < todo.length ) {

position = priority.indexOf( index+1 );

todoInOrderOfPriority.push( todo[position] );

index++;

}

alert("Previous order:\n\n" + todo + "\n\n"

+ "Now in order of priority:\n\n" + todoInOrderOfPriority);

</script>

</head>

</html>

**Without running the code**you need to say whether the code will correctly show the items stored in*todo*in the order indicated by the numbers stored in *priority*.



No, the code does not show the items in *todo* in the correct order of priority.



Yes, the code shows the items in *todo* in the correct order of priority. **OK**

2.

The task in this question is essentially the same as the last question. However, the code is not the same as the last question.

Tasks are stored in the array *todo* and the priority of those items are stored in array *priority*. 1 means the highest priority, 2 means the second highest priority, and so on. The code will then display the items in order from the highest priority item to the lowest priority item.

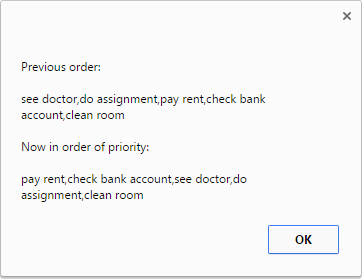
For example, if the data is stored like this:

var todo = ["see doctor", "do assignment", "pay rent",

"check bank account", "clean room"];

var priority = [3, 4, 1, 2, 5];

then the code needs to shows this result:



Here is the code. As mentioned, this code is **not the same**as the code in the previous question.

<!doctype html>

<html>

<head>

<script>

var todo = ["see doctor", "do assignment", "pay rent",

"check bank account", "clean room"];

var priority = [3, 4, 1, 2, 5];

var index = 0;

var todoInOrderOfPriority = [];

todoInOrderOfPriority.length = todo.length;

while ( index < todo.length ) {

position = priority.indexOf( index+1 );

todoInOrderOfPriority[index] = todo[position];

index++;

}

alert("Previous order:\n\n" + todo + "\n\n"

+ "Now in order of priority:\n\n" + todoInOrderOfPriority);

</script>

</head>

</html>

**Without running the code**you need to say whether the code will correctly show the items stored in*todo* in the order indicated by the numbers stored in *priority*.



No, the code does not show the items in *todo*in the correct order of priority.



Yes, the code shows the items in *todo*in the correct order of priority. **OK**

3.

Someone will purchase a particular product if one or more of these two situations if true.

1) if it is high quality and expensive, they will purchase the product.

2) If it is cheap (regardless of the quality), they will purchase the product

JavaScript variables are created as shown below.

var high\_quality;

var cheap;

var will\_purchase;

Using these variables, a programmer writes one line of code which correctly encapsulates the logic mentioned above. Which one of the following lines of code correctly represents the logic shown above?



will\_purchase = (high\_quality & !cheap) || (!high\_quality && cheap);



will\_purchase = (high\_quality && !cheap) || cheap;

 **OK**

will\_purchase = (high\_quality && cheap) || !cheap;



will\_purchase = high\_quality || (!high\_quality && cheap);

3 questions unanswered