Started on	Wednesday, 30 August 2023, 7:25 PM
State	Finished
Completed on	Friday, 1 September 2023, 7:26 PM
Time taken	2 days
Grade	**8.00** out of 8.00 (**100**%)

QUESTION 1

Correct

Mark 1.00 out of 1.00

Implement the *getModifiedArray(array)* function, which takes an arbitrary *array*, and returns an array with the value of the first element of the array equal to "Start", the last element of the array equal to "End" and the rest of elements should be the same as in an initial array. The initial array should stay unchanged.

Function example:

getModifiedArray([12, 6, 22, 0, -8])); // ['Start', 6, 22, 0, 'End']

* For correct passing of all tests don't use console.log() method in your code.

Answer: (penalty regime: 0 %)

Reset answer

```
function getModifiedArray(array) {
  const copiedArray = array.slice();
  copiedArray[0] = 'Start';
  copiedArray[copiedArray.length - 1] = 'End';
  return copiedArray;
}
```

	Test	Expected	Got	
✔	`console.log(getModifiedArray([12, 6, 22, 0, -8]));`	`[ 'Start', 6, 22, 0, 'End' ]`	`[ 'Start', 6, 22, 0, 'End' ]`	✔
✔	`console.log(getModifiedArray(["Kate", "Peter", "Mark", "Sam"]));`	`[ 'Start', 'Peter', 'Mark', 'End' ]`	`[ 'Start', 'Peter', 'Mark', 'End' ]`	✔
✔	`console.log(getModifiedArray([false, 10, 'mail', true, 20, 30]));`	`[ 'Start', 10, 'mail', true, 20, 'End' ]`	`[ 'Start', 10, 'mail', true, 20, 'End' ]`	✔
✔	`console.log(getModifiedArray([100, 200]));`	`[ 'Start', 'End' ]`	`[ 'Start', 'End' ]`	✔

✔	`const arr = [false, 10, 'mail', true, 20, 30];`
	`getModifiedArray(arr)`
	`console.log(arr);`
		`[ false, 10, 'mail', true, 20, 30 ]`	`[ false, 10, 'mail', true, 20, 30 ]`	✔

✔	`const arr = [100, 200];`
	`getModifiedArray(arr)`
	`console.log(arr);`
		`[ 100, 200 ]`	`[ 100, 200 ]`	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

QUESTION 2

Correct

Mark 1.00 out of 1.00

The function *filterByN* receives an array of integers, a *number* and a *parameter* (*greater, less*). Print a new array, where all elements will be greater/less than this number

By default, the *number* is *0*, the *parameter* is *greater*.

Example:

filterNums([-1, 2, 4, 0, 55, -12, 3], 11, 'greater'); //[55]

filterNums([-2, 2, 3, 0, 43, -13, 6], 6, 'less'); // [-2, 2, 3, 0, -13]

filterNums([-2, 2, 3, 0, 43, -13, 6], -33, 'less'); // []

filterNums([-2, 2, 3, 0, 43, -13, 6]); // [2, 3, 43, 6]

filterNums([-2, 2, 3, 0, 43, -13, 6], 23); // [43]

* For correct passing of all tests don't use console.log() method in your code.

Answer: (penalty regime: 0 %)

Reset answer

```
const filterNums = (array, number = 0, param = 'greater') => {
  let filteredArr;

  if (param === 'greater') {
    filteredArr = array.filter((el) => el > number);
  } else if (param === 'less') {
    filteredArr = array.filter((el) => el < number);
  }
  return filteredArr;
};
```

	Test	Expected	Got	
✔	console.log(filterNums([-3, 3, 4, 0, 44, -11, 5], 11, 'greater'));	[44]	[44]	✔
✔	console.log(filterNums([-3, 3, 4, 0, 44, -11, 5], 5, 'less'));	[-3, 3, 4, 0, -11]	[-3, 3, 4, 0, -11]	✔
✔	console.log(filterNums([-3, 3, 4, 0, 44, -11, 5], -30, 'less'));	[]	[]	✔
✔	console.log(filterNums([-3, 3, 4, 0, 44, -11, 5]));	[3, 4, 44, 5]	[3, 4, 44, 5]	✔
✔	console.log(filterNums([-3, 3, 4, 0, 44, -11, 5], 9));	[44]	[44]	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

QUESTION 3

Correct

Mark 1.00 out of 1.00

Find the maximum interval between two consecutive arguments.

Example:

maxInterv(3, 5, 2, 7); //5
maxInterv(3, 5, 2, 7, 11, 0, -2); //11
maxInterv(3, 5); //2
maxInterv(3); //0

* For correct passing of all tests don't use console.log() method in your code.

Answer: (penalty regime: 0 %)

Reset answer

```
function maxInterv(...args) {
  if (args.length < 2) {
    return 0;
  }
  let maxInterval = -Infinity;
  for (let i = 1; i < args.length; i++) {
    const interval = Math.abs(args[i] - args[i - 1]);
    if (interval > maxInterval) {
      maxInterval = interval;
    }
  }

  return maxInterval;
}
```

	Test	Expected	Got	
✔	console.log(maxInterv(3, 5, 2, 7))	5	5	✔
✔	console.log(maxInterv(3, 5, 2, 7, 11, 0, -2))	11	11	✔
✔	console.log(maxInterv(3, 5))	2	2	✔
✔	console.log(maxInterv(3));	0	0	✔
✔	console.log(maxInterv(3, 5, 2, 8));	6	6	✔
✔	console.log(maxInterv(3, 5, 2, 37, 11, 0, -2))	35	35	✔
✔	console.log(maxInterv(8));	0	0	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

QUESTION 4

Correct

Mark 1.00 out of 1.00

The function takes any number of strings and returns the sum of their lengths.

Example:

```
console.log(sumOfLen('hello', 'hi')); //7
console.log(sumOfLen('hi')); //2
console.log(sumOfLen()); //0
console.log(sumOfLen('hello', 'hi', 'my name', 'is')); //16
```

* For correct passing of all tests don't use console.log() method in your code.

Answer: (penalty regime: 0 %)

Reset answer

```
const sumOfLen = (...strings) => {
    const str = strings.join('');
    return str.length;
}
```

	Test	Expected	Got	
✔	`console.log(sumOfLen('hello', 'hi'));`	7	7	✔
✔	`console.log(sumOfLen('hi'));`	2	2	✔
✔	`console.log(sumOfLen());`	0	0	✔
✔	`console.log(sumOfLen('hello', 'hi', 'my name', 'is'));`	16	16	✔
✔	`console.log(sumOfLen('hello', 'hi', 'my name', 'is2'));`	17	17	✔
✔	`console.log(sumOfLen('hello', 'my name', 'is'));`	14	14	✔
✔	`console.log(sumOfLen('hello', 'my name'));`	12	12	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

QUESTION 5

Correct

Mark 1.00 out of 1.00

Write a function *combineArray(arr1, arr2)*, which takes 2 arrays, and returns a new array consisting only of numeric elements of arrays *arr1* and *arr2*.

Function example:

combineArray([12, "User01", 22, true, -8], ["Index", 6, null, 15])); // [12, 22, -8, 6, 15]

* For correct passing of all tests don't use console.log() method in your code.

Answer: (penalty regime: 0 %)

Reset answer

```
function combineArray(arr1, arr2) {
  let newArr = arr1.concat(arr2);
  let filtered = newArr.filter((el) => {
    if (typeof el === 'number') {
      return el;
    }
  });
  return filtered;
}
```

	Test	Expected	Got	
✔	`console.log(combineArray([12, "User01", 22, true, -8], ["Index", 6, null, 15]));`	`[ 12, 22, -8, 6, 15 ]`	`[ 12, 22, -8, 6, 15 ]`	✔
✔	`console.log(combineArray(["User01", "User02", "User03", "User04"], ["Data1", 33, "Data2", 44]));`	`[ 33, 44 ]`	`[ 33, 44 ]`	✔
✔	`console.log(combineArray([10, 20, 30], ["Data1", "Data2", "Data3", "Data4", "Data5"]));`	`[ 10, 20, 30 ]`	`[ 10, 20, 30 ]`	✔
✔	`console.log(combineArray([1, 2, 3, 4, 5], [6, 7, 8, 9, 10]));`	`[ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ]`	`[ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ]`	✔
✔	`console.log(combineArray(['1', '2', '3', '4' ], ['first', 'second', 'third']));`	`[]`	`[]`	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

QUESTION 6

Correct

Mark 1.00 out of 1.00

Implement the *longestLogin(loginList)* function, which takes an array of user logins *loginList* and returns the longest login. If the logins of the same length are the longest in the array, the login element with the largest index is returned. Tip: You can use the reduce() method to solve the task.

Function examples:

longestLogin(["serg22", "tester_2", "Prokopenko", "guest"]); // Prokopenko

longestLogin(["user1", "user2", "333", "user4", "aa"]); // user4

* For correct passing of all tests don't use console.log() method in your code.

Answer: (penalty regime: 0 %)

Reset answer

```
function longestLogin(loginList) {
  if (loginList.length === 0) {
    return null;
  }
  return loginList.reduce((str1, str2) => {
    if (
      str2.length > str1.length ||
      (str2.length === str1.length &&
        loginList.indexOf(str2) > loginList.indexOf(str1))
    ) {
      return str2;
    }
    return str1;
  });
}
```

	Test	Expected	Got	
✔	console.log(longestLogin(["maxxx", "NewUser", "admin111" , "Administrator"]));	Administrator	Administrator	✔
✔	console.log(longestLogin(["User123", "Steven Dobson", "qwerty12345"]));	Steven Dobson	Steven Dobson	✔
✔	console.log(longestLogin(["Carl1999", "ivan@gmail.com", "nick-name"]));	ivan@gmail.com	ivan@gmail.com	✔
✔	console.log(longestLogin(["user1", "user2", "333", "user4", "aa"]));	user4	user4	✔
✔	console.log(longestLogin(["larian", "questttt", "longest_user_name", "Nick Nickson"]));	longest_user_name	longest_user_name	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

QUESTION 7

Correct

Mark 1.00 out of 1.00

Implement the *processArray(arr, factorial)* function, which takes the first parameter of the array *arr*, and the second parameter the function *factorial* and processes each element of the array *arr* with the function *factorial*, returning a new array (the source array *arr* does not change)

The function *factorial(n)* calculates and returns the factorial of the number *n*. For example factorial(4) returns 24.

Example

// determines the factorial of the number n

function factorial(n) { // your code};

processArray([1, 2, 3, 4, 5], factorial); // [1, 2, 6, 24, 120]

* For correct passing of all tests don't use console.log() method in your code.

Answer: (penalty regime: 0 %)

Reset answer

```
function factorial(n) {
  if (n === 0 || n === 1) {
    return 1;
  } else {
    return n * factorial(n - 1);
  }
}

function processArray(arr, factorial) {
  const result = arr.map((el) => factorial(el));
  return result;
}
```

	Test	Expected	Got	
✔	`console.log(processArray([1, 2, 3, 4, 5, 6], factorial));`	`[ 1, 2, 6, 24, 120, 720 ]`	`[ 1, 2, 6, 24, 120, 720 ]`	✔
✔	`console.log(processArray([6, 5, 4, 3, 2, 1], factorial));`	`[ 720, 120, 24, 6, 2, 1 ]`	`[ 720, 120, 24, 6, 2, 1 ]`	✔
✔	`console.log(processArray([0, 9, 4, 12], factorial));`	`[ 1, 362880, 24, 479001600 ]`	`[ 1, 362880, 24, 479001600 ]`	✔
✔	`console.log(processArray([9, 8, 13, 22], factorial));`	`[ 362880, 40320, 6227020800, 1.1240007277776077e+21 ]`	`[ 362880, 40320, 6227020800, 1.1240007277776077e+21 ]`	✔
✔	`console.log(processArray([], factorial));`	`[]`	`[]`	✔

✔	`const arr = [2, 4, 6];`
	`console.log(processArray(arr, factorial));`
	`console.log(arr);`
		`[ 2, 24, 720 ]`
		`[ 2, 4, 6 ]`
			`[ 2, 24, 720 ]`
			`[ 2, 4, 6 ]`
				✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

QUESTION 8

Correct

Mark 1.00 out of 1.00

Using the default parameter technique, overload the *overloadedFunc()* function, which takes 3 arguments. For the 1st argument of the function set the default value [1, 2, 3], for the 2nd - the value 2, for the 3rd - the function that returns the product of the first two arguments, and the function can multiply both arrays and numbers.

The *overloadedFunc()* function returns the result of the default function.

For example:

Test	Result
`console.log(overloadedFunc());`	`[ 2, 4, 6 ]`
`console.log(overloadedFunc([2,4,6,8]));`	`[ 4, 8, 12, 16 ]`
`console.log(overloadedFunc([2,4,6], 3));`	`[ 6, 12, 18 ]`
`console.log(overloadedFunc(10));`	`20`
`console.log(overloadedFunc(8, 3));`	`24`

Answer: (penalty regime: 0 %)

Reset answer

```
function overloadedFunc(a = [1, 2, 3], b = 2, c = (x, y) => x * y) {
  if (!Array.isArray(a)) {
    return c(a, b);
  }
  if (Array.isArray(a)) {
    return a.map((el) => c(el, b));
  }
}
```

	Test	Expected	Got	
✔	`console.log(overloadedFunc());`	`[ 2, 4, 6 ]`	`[ 2, 4, 6 ]`	✔
✔	`console.log(overloadedFunc([2,4,6,8]));`	`[ 4, 8, 12, 16 ]`	`[ 4, 8, 12, 16 ]`	✔
✔	`console.log(overloadedFunc([2,4,6], 3));`	`[ 6, 12, 18 ]`	`[ 6, 12, 18 ]`	✔
✔	`console.log(overloadedFunc(10));`	`20`	`20`	✔
✔	`console.log(overloadedFunc(8, 3));`	`24`	`24`	✔
✔	`console.log(overloadedFunc(undefined, undefined, func.sum));`	`[ 3, 4, 5 ]`	`[ 3, 4, 5 ]`	✔
✔	`console.log(overloadedFunc(3, undefined, func.sum));`	`5`	`5`	✔
✔	`console.log(overloadedFunc(10, 5, func.sum));`	`15`	`15`	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

◄ Video. Functions for work with arrays [14:59]

Jump to...

Tasks * JS for React ►