H

Full Name:
Email:
Test Name:
Taken On:
Time Taken:
Work Experience:
Invited by:
Invited on:
Tags Score:

Kenneth Choi kennethichoi@gmail.com MBA: JavaScript 1 Aug 2019 15:05:00 PDT 44 min 55 sec/ 45 min 1 years Jeff 1 Aug 2019 15:01:13 PDT Advanced 36.25/60 Callbacks 30/65 Closure 35/35 Currying 0/40 Essential 25/25 Javascript 131.25/230 Javascript Async/Await 5/5 Javascript Asynchronous 5/10

Javascript Context 6.25/10

Javascript ES6 15/20

Javascript Hoisting 5/10

Javascript Modules 10/10

Javascript Promises 15/25

Javascript Scope 10/15

Javascript Strict Mode 1.25/5

Javascript General Knowledge 20/20

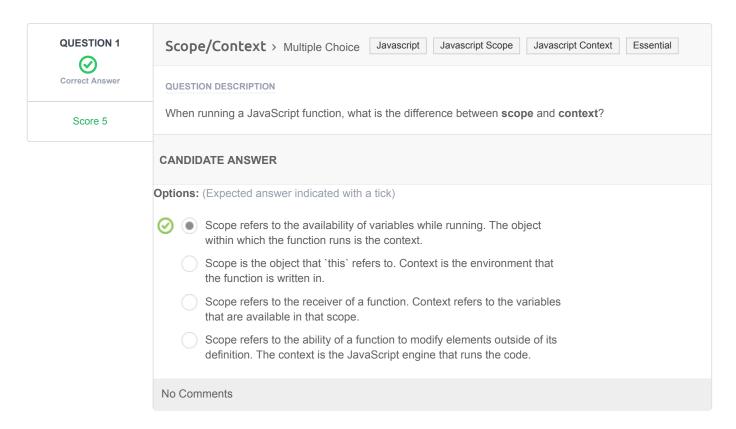
57.1% 131/230

scored in **MBA: JavaScript** in 44 min 55 sec on 1 Aug 2019 15:05:00 PDT

Recruiter/Team Comments:

	Question Description	Time Taken	Score	Status
Q1	Scope/Context > Multiple Choice	20 sec	5/ 5	Ø
Q2	Modules > Multiple Choice	48 sec	5/ 5	Ø
Q3	Modules > Multiple Choice	33 sec	5/ 5	Ø
Q4	JavaScript Basics > Multiple Choice	13 sec	5/ 5	⊘
Q5	JavaScript Basics > Multiple Choice	38 sec	5/ 5	⊘

Q6	Fill in the Blank > Multiple Choice	26 sec	5/ 5	Ø
Q7	Closure > Coding	12 min 4 sec	35/ 35	⊘
Q8	Currying Functions > Coding	14 min 33 sec	0/ 40	8
Q9	Binding > Coding	5 min 48 sec	0/ 35	8
Q10	Function that accepts a callback > Coding	57 sec	30/ 30	⊘
Q11	Promises > Multiple Choice	58 sec	5/ 5	⊘
Q12	Promises/Async > Multiple Choice	33 sec	0/5	8
Q13	Promises > Multiple Choice	1 min 7 sec	5/ 5	⊘
Q14	Promises > Multiple Choice	29 sec	0/5	8
Q15	Promises > Multiple Choice	29 sec	5/ 5	⊘
Q16	Strict Mode > Multiple Choice	1 min 20 sec	1.25/ 5	⊘
Q17	Scope > Multiple Choice	1 min 10 sec	5/ 5	⊘
Q18	Scope > Multiple Choice	1 min 1 sec	0/5	8
Q19	Asynchronous JavaScript > Multiple Choice	28 sec	5/ 5	⊘
Q20	Numbers > Multiple Choice	37 sec	5/ 5	⊘
Q21	Async/Await > Multiple Choice	32 sec	5/ 5	⊘
Q22	ES6 > Multiple Choice	1 sec	0/5	8





Score 5

Modules > Multiple Choice | Javascript | Javascript Modules | Essential

QUESTION DESCRIPTION

Given the following directory structure:

```
main-directory
  |__ index.js
 |__ util
       |__ add.js
```

```
// ******
// add.js
// ******
export const five = (num) => {
 return num + 5;
};
export const ten = (num) => {
 return num + 10;
};
```

What is missing from the following code to allow index.js to successfully access and use the five function from add.js?

```
// index.js
// ******
// Missing code goes here
// *****
const a = add.five(0);
const b = add.five(10);
return a + b;
```

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)



import { five } from './util/add.js';

- import add from './util';
 - import util from './util/add.js';
 - import { ten } from './util';

QUESTION 3 Modules > Multiple Choice | Javascript | Javascript Modules (Correct Answer **QUESTION DESCRIPTION** Given the following project structure: Score 5 main-directory |__ index.js |__ compound_string.js What is missing from the following files in order to allow index.js to successfully access and use code from compound_string.js? // index.js // ****** // Missing code goes here const result = new CompoundString('Test string'); return result; // compound string.js export default class CompoundString { constructor(str) { this.str = str; **CANDIDATE ANSWER Options:** (Expected answer indicated with a tick)

QUESTION 4	JavaScript Basics > Multiple Choice
Correct Answer	QUESTION DESCRIPTION
Score 5	V8, SpiderMonkey, JavaScriptCore/Nitro, and Chakra are all examples of
	CANDIDATE ANSWER
	Options: (Expected answer indicated with a tick) web browsers JavaScript compilers JavaScript engines JavaScript frameworks
	No Comments
QUESTION 5 Correct Answer	JavaScript Basics > Multiple Choice
Score 5	QUESTION DESCRIPTION Which of the following are primitives in JavaScript (ES2015)?
	CANDIDATE ANSWER
	Options: (Expected answer indicated with a tick)
	StringsNumbersFunctions
	onull Array
	✓ ● Symbols✓ ● Booleans
	Objects undefined



Score 5

to make this function work as intended.

Essential

QUESTION DESCRIPTION

The following JavaScript function should return the highest integer in an array. Fill in the missing line of code

```
function findHighest(arr) {
 // *****
  // Missing code goes here
 for (let i = 1; i < arr.length; i++) {</pre>
    if (arr[i] > highestNum) highestNum = arr[i];
 return highestNum;
}
```

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)



- let highestNum = arr[0];
 - let highestNum = 0;
 - const highestNum = arr[0];
 - const highestNum = 0;



Score 35

Closure > Coding Closure Javascript

QUESTION DESCRIPTION

Write a function called **chantCreator**.

chantCreator should create a **chant** function which will store words to be chanted and return an array with all previously called words.

Example:

```
const chant = chantCreator();
chant('a'); // returns ['a']
chant('b'); // returns ['a', 'b']
chant('c'); // returns ['a', 'b', 'c']
```

INTERNAL NOTES

35

CANDIDATE ANSWER

Language used: JavaScript (Node.js)

```
1 /*
2 * Complete the function below.
3 */
4 function chantCreator() {
5    let values = [];
6    return function chant(char) {
7      values.push(char);
8      return values;
9    }
10 }
11
12
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
1 Word	Easy	Sample case	Success	5	0.088 sec	35.9 KB
2 Words	Easy	Sample case	Success	10	0.0804 sec	35.7 KB
Testcase 3	Medium	Sample case	Success	10	0.103 sec	36.4 KB
Many Calls	Medium	Hidden case	Success	10	0.0854 sec	36.6 KB



Score 0

Currying Functions > Coding Javascript Currying

QUESTION DESCRIPTION

Write a function **myCurry** which takes in a callback, and the number of arguments to take before executing, and returns a curried function that accepts **one** argument.

Once you have reached the specified number of arguments, return and invoke the callback with the **array** of collected arguments.

Example:

```
const sum = function(array) {
  return array.reduce((a, b) => a + b);
};

const curriedSum = myCurry(sum, 3);

const stepOne = curriedSum(1); // returns a function
  const stepTwo = stepOne(2); // returns a function
  const stepThree = stepTwo(3); // returns 6
```

INTERNAL NOTES

40

CANDIDATE ANSWER

Language used: JavaScript (Node.js)

```
1 /*
 2 * Complete the function below.
 3 */
 4 function myCurry(func, num) {
     let args = [];
 6
     return function _myCurry(newArg) {
         args.push(newArg);
 8
         if (args.length === num) {
              return func.apply(this, args);
         } else {
              return _myCurry;
         }
      }
14 }
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Example Test	Easy	Sample case	Runtime Error	0	0.0861 sec	36.8 KB
Negative	Medium	Sample case	Runtime Error	0	0.0767 sec	37.1 KB
Many Arguments	Medium	Sample case	Runtime Error	0	0.0897 sec	36.6 KB



Score 0

Binding > Coding Javascript Callbacks

QUESTION DESCRIPTION

Write your own myBind(ctx) method. Your function should be able to accept bind time and call time arguments and work like the built in bind method.

Example:

```
class Cat {
 constructor(name) {
   this.name = name;
 says(sound, person) {
   console.log(`${this.name} says ${sound} to ${person}!`);
   return true;
}
class Dog {
 constructor(name) {
    this.name = name;
const jet = new Cat("Jet");
const pavlov = new Dog("Pavlov");
const myBoundSays = jet.says.myBind(pavlov);
const BoundSays = jet.says.bind(pavlov)
myBoundSays("meow", "a tree"); // Pavlov says meow to a tree!
BoundSays("meow", "a tree"); // Pavlov says meow to a tree!
```

INTERNAL NOTES

40

CANDIDATE ANSWER

Language used: JavaScript (Node.js)

```
1  /*
2  * Complete the 'myBind' function below..
3  */
4
5  Function.prototype.myBind = function(context, bindArgs) {
6    return this.call(context, ...bindArgs, ...arguments);
7
8 }
9
10
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Binds Context Correctly	Easy	Sample	Runtime	0	0.0781 sec	36.8 KB

Bind and Call Time Medium Sample & Runtime 0 0.0824 sec 36.4 KB
Arguments Error



Score 30

QUESTION DESCRIPTION

Create a function myMap that takes in an array and an optional callback. myMap will pass each element of the array into the callback and return an array with the return values of the callback. If no callback is provided, return a shallow copy of the original array.

Example:

```
const exampleCallback = (el) => el + 5;
const arr = [10, 20, 30, 40];
myMap(arr, exampleCallback); // returns [15, 25, 35, 45]
const shallowCopy = myMap(arr); // returns [10, 20, 30, 40]
shallowCopy[0] = 5;
                               // [5, 20, 30, 40]
shallowCopy
                               // [10, 20, 30, 40]
arr
```

INTERNAL NOTES

30

CANDIDATE ANSWER

Language used: JavaScript (Node.js)

```
1 /*
2 * Complete the function below.
4 function myMap(arr, cb) {
      if (!cb) cb = el => el;
      let output = [];
      arr.forEach(el => {
          output.push(cb(el));
     })
      return output;
13 }
14
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Example Test	Easy	Sample case	Success	10	0.0874 sec	37 KB
More arguments	Medium	Sample case	Success	20	0.0771 sec	35.9 KB

QUESTION 11	Promises > Multiple Choice					
Correct Answer	QUESTION DESCRIPTION					
Score 5	Which of the following statements about ES6 Promise objects are true?					
	CANDIDATE ANSWER					
	Options: (Expected answer indicated with a tick)					
	Promises allow the writing of asynchronous JavaScript code in a linear fashion.					
	Promises avoid the need for deeply nested callbacks for asynchronous operations.					
	Promises prevent asynchronous functions from executing and transforms them into synchronous functions.					
	Promises create a private scope around a function, preventing it from making changes to surround variables.					
	No Comments					



Score 0

Promises/Async > Multiple Choice | Javascript | Javascript Promises | Advanced

QUESTION DESCRIPTION

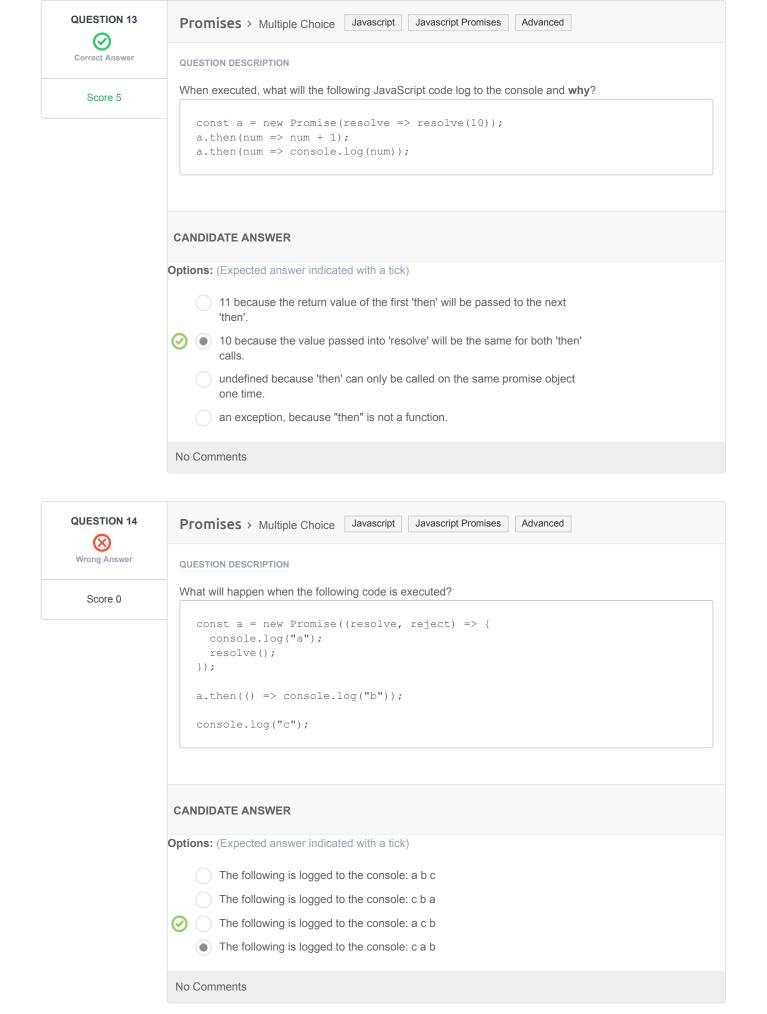
What will happen when the following code is executed?

```
const a = () => new Promise((resolve, reject) => {
 setTimeout(() => {
   resolve("a");
 }, 5000);
});
const b = () => new Promise((resolve, reject) => {
 setTimeout(() => {
   resolve("b");
 }, 5000);
});
async function asyncTest() {
 await a();
 await b();
asyncTest().then(() => console.log('ASYNC'));
```

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- "ASYNC" is logged immediately.
- "ASYNC" is logged after 5 seconds.
- "ASYNC" is logged after 10 seconds.
 - Nothing is logged to the console.





Score 5

Advanced

QUESTION DESCRIPTION

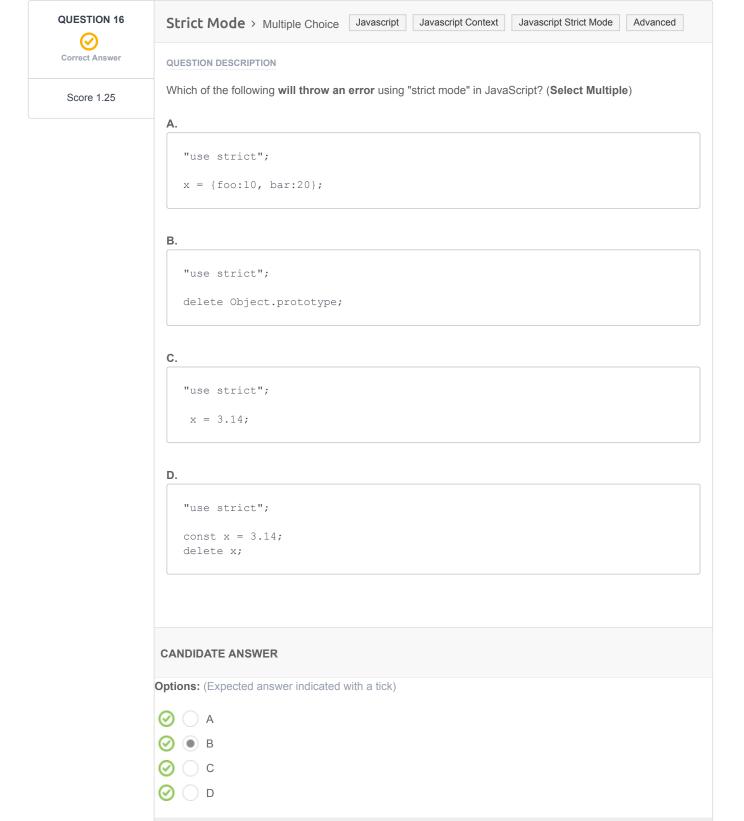
What will happen when the following code is executed?

```
const a = () => new Promise((resolve, reject) => {
 setTimeout(() => {
   resolve("a");
 }, 5000);
});
const b = () => new Promise((resolve, reject) => {
 setTimeout(() => {
   resolve("b");
 }, 5000);
});
function promiseTest() {
 Promise.all([a(), b()])
   .then(() => console.log('PROMISE'));
promiseTest();
```

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- "PROMISE" is logged immediately.
- "PROMISE" is logged after 5 seconds.
 - "PROMISE" is logged after 10 seconds.
 - An exception is thrown.





Score 5

QUESTION DESCRIPTION

The following code defines a global variable 'a' outside of a function. It defines a local variable 'a' inside the function, using **let**. What will happen when this code is executed?

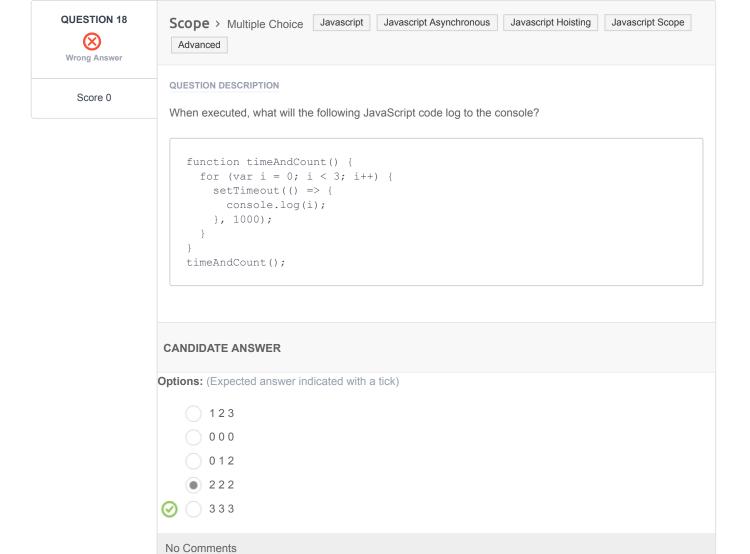
```
a = 5;
function scopeTest() {
  console.log(a);

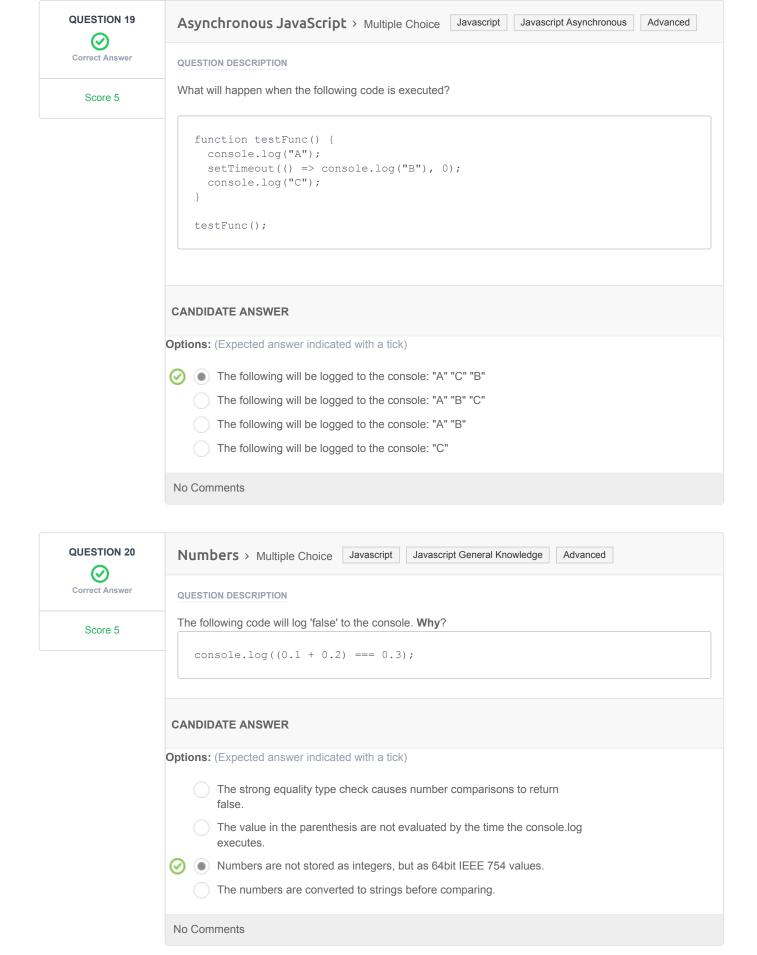
let a = 6;
}
scopeTest();
```

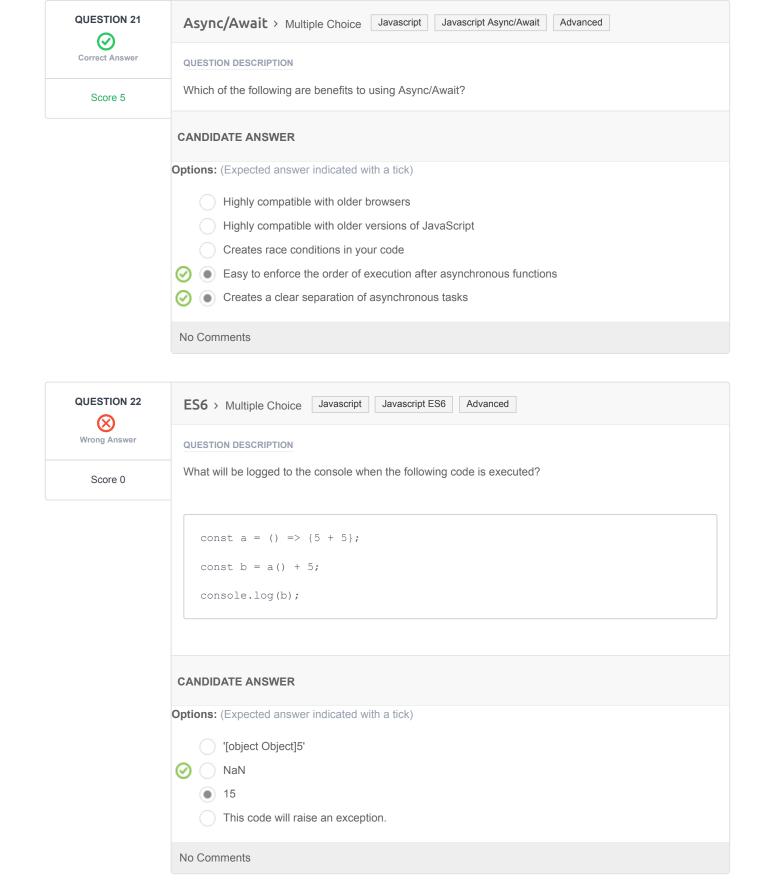
CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- It will log 'undefined' because the local variable 'a' will be hoisted to the beginning of the function, but it has not been defined where the 'console.log' is called.
- It will log '5' because the local 'a' has not been initialized, and the 'a' being referenced by 'console.log' is the global 'a'.
- It will log '6' because the local 'a' and its definition will be hoisted to the beginning of the function, overwriting access to the original global 'a'.
- An exception will be raised because the variable name 'a' will be reserved at the top of the function, overwriting access to the original global 'a', but the local 'a' has not been initialized where the 'console.log' is called.







PDF generated at: 1 Aug 2019 22:51:15 UTC