MBA: JavaScript > kennethichoi@gmail.com

Full Name:	Kenneth Choi
Email:	kennethichoi@gmail.com
Test Name:	**MBA: JavaScript**
Taken On:	1 Aug 2019 15:05:00 PDT
Time Taken:	44 min 55 sec/ 45 min
Work Experience:	1 years
Invited by:	Jeff
Invited on:	1 Aug 2019 15:01:13 PDT

57.1%
131/230

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

Tags Score:

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 General Knowledge	20/20
Javascript Hoisting	5/10
Javascript Modules	10/10
Javascript Promises	15/25
Javascript Scope	10/15
Javascript Strict Mode	1.25/5

Recruiter/Team Comments:

No Comments.

	Question Description	Time Taken	Score	Status
Q1	Scope/Context > **Multiple Choice**	20 sec	5/ 5	✓
Q2	Modules > **Multiple Choice**	48 sec	5/ 5	✓
Q3	Modules > **Multiple Choice**	33 sec	5/ 5	✓
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	✓
Q7	**Closure** › **Coding**	12 min 4 sec	35/ 35	✓
Q8	**Currying Functions** › **Coding**	14 min 33 sec	0/ 40	✗
Q9	**Binding** › **Coding**	5 min 48 sec	0/ 35	✗
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	✗
Q13	**Promises** › **Multiple Choice**	1 min 7 sec	5/ 5	✓
Q14	**Promises** › **Multiple Choice**	29 sec	0/ 5	✗
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	✗
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	✗

QUESTION 1

✓ Correct Answer

Score 5

Scope/Context › Multiple Choice

Javascript | Javascript Scope | Javascript Context | Essential

QUESTION DESCRIPTION

When running a JavaScript function, what is the difference between **scope** and **context**?

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- ✓ ● Scope refers to the availability of variables while running. The object within which the function runs is the context.
- ○ Scope is the object that `this` refers to. Context is the environment that the function is written in.
- ○ Scope refers to the receiver of a function. Context refers to the variables that are available in that scope.
- ○ Scope refers to the ability of a function to modify elements outside of its definition. The context is the JavaScript engine that runs the code.

No Comments

QUESTION 2

✓ Correct Answer

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';

No Comments

QUESTION 3

✓ Correct Answer

Score 5

Modules › Multiple Choice

Javascript | Javascript Modules | Essential

QUESTION DESCRIPTION

Given the following project structure:

```
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)

- ○ const CompoundString = require('./compound_string.js');
- ○ require('./compound_string.js');
- ○ module.import('./compound_string.js');
- ○ import { CompoundString } from './compound_string.js';
- ✓ ● import CompoundString from './compound_string.js';

No Comments

QUESTION 4

✓ Correct Answer

Score 5

JavaScript Basics › Multiple Choice

Javascript | Javascript General Knowledge

QUESTION DESCRIPTION

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

Score 5

JavaScript Basics › Multiple Choice

Javascript | Javascript ES6 | Javascript General Knowledge | Essential

QUESTION DESCRIPTION

Which of the following are primitives in JavaScript (ES2015)?

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- ✓ ● Strings
- ✓ ● Numbers
- ○ Functions
- ✓ ● null
- ○ Array
- ✓ ● Symbols
- ✓ ● Booleans
- ○ Objects
- ✓ ● undefined

No Comments

QUESTION 6

✓ Correct Answer

Score 5

Fill in the Blank › Multiple Choice

Javascript | Javascript ES6 | Javascript General Knowledge | Essential

QUESTION DESCRIPTION

The following JavaScript function should return the highest integer in an array. Fill in the missing line of code to make this function work as intended.

```
function findHighest(arr) {

  // *******
  // Missing code goes here
  // *******

  for (let i = 1; i < arr.length; i++) {
    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;

No Comments

QUESTION 7

✓ Correct Answer

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)**

```
/*
 * Complete the function below.
 */
function chantCreator() {
    let values = [];
    return function chant(char) {
        values.push(char);
        return values;
    }
}


```

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

No Comments

QUESTION 8

Wrong Answer

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)**

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


```

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

No Comments

QUESTION 9

Wrong Answer

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 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)**

```
/*
 * Complete the 'myBind' function below..
 */

Function.prototype.myBind = function(context, bindArgs) {
    return this.call(context, ...bindArgs, ...arguments);

}


```

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

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

No Comments

QUESTION 10

✓ Correct Answer

Score 30

Function that accepts a callback › Coding

Javascript Callbacks

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;

shallowCopy                     // [5, 20, 30, 40]
arr                             // [10, 20, 30, 40]
```

INTERNAL NOTES

30

CANDIDATE ANSWER

Language used: **JavaScript (Node.js)**

```
/*
 * Complete the function below.
 */
function myMap(arr, cb) {
    if (!cb) cb = el => el;

    let output = [];
    arr.forEach(el => {
        output.push(cb(el));
    })

    return output;
}


```

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

No Comments

QUESTION 11

✓ Correct Answer

Score 5

Promises › Multiple Choice

Javascript | Javascript Promises | Advanced

QUESTION DESCRIPTION

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

QUESTION 12

✗ Wrong Answer

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.

No Comments

QUESTION 13

✓ Correct Answer

Score 5

Promises › Multiple Choice

Javascript | Javascript Promises | Advanced

QUESTION DESCRIPTION

When executed, what will the following JavaScript code log to the console and **why**?

```
const a = new Promise(resolve => resolve(10));
a.then(num => num + 1);
a.then(num => console.log(num));
```

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- ○ 11 because the return value of the first 'then' will be passed to the next 'then'.
- ✓ ● 10 because the value passed into 'resolve' will be the same for both 'then' calls.
- ○ undefined because 'then' can only be called on the same promise object one time.
- ○ an exception, because "then" is not a function.

No Comments

QUESTION 14

✗ Wrong Answer

Score 0

Promises › Multiple Choice

Javascript | Javascript Promises | Advanced

QUESTION DESCRIPTION

What will happen when the following code is executed?

```
const a = new Promise((resolve, reject) => {
  console.log("a");
  resolve();
});

a.then(() => console.log("b"));

console.log("c");
```

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- ○ The following is logged to the console: a b c
- ○ The following is logged to the console: c b a
- ✓ ○ The following is logged to the console: a c b
- ● The following is logged to the console: c a b

No Comments

QUESTION 15

✓ Correct Answer

Score 5

Promises › 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);
});

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.

No Comments

QUESTION 16

✓ Correct Answer

Score 1.25

Strict Mode › Multiple Choice

Javascript | Javascript Context | Javascript Strict Mode | Advanced

QUESTION DESCRIPTION

Which of the following **will throw an error** using "strict mode" in JavaScript? (**Select Multiple**)

A.

```
"use strict";

x = {foo:10, bar:20};
```

B.

```
"use strict";

delete Object.prototype;
```

C.

```
"use strict";

 x = 3.14;
```

D.

```
"use strict";

const x = 3.14;
delete x;
```

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

✓ ○ A

✓ ● B

✓ ○ C

✓ ○ D

No Comments

QUESTION 17

✓ Correct Answer

Score 5

Scope › Multiple Choice   Javascript   Javascript ES6   Javascript Scope   Javascript Hoisting   Advanced

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.

No Comments

QUESTION 18

Wrong Answer

Score 0

Scope › Multiple Choice

Javascript | Javascript Asynchronous | Javascript Hoisting | Javascript Scope | Advanced

QUESTION DESCRIPTION

When executed, what will the following JavaScript code log to the console?

```
function timeAndCount() {
  for (var i = 0; i < 3; i++) {
    setTimeout(() => {
      console.log(i);
    }, 1000);
  }
}
timeAndCount();
```

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- ○ 1 2 3
- ○ 0 0 0
- ○ 0 1 2
- ● 2 2 2
- ✓ ○ 3 3 3

No Comments

QUESTION 19

✓ Correct Answer

Score 5

Asynchronous JavaScript › Multiple Choice

Javascript | Javascript Asynchronous | Advanced

QUESTION DESCRIPTION

What will happen when the following code is executed?

```
function testFunc() {
  console.log("A");
  setTimeout(() => console.log("B"), 0);
  console.log("C");
}

testFunc();
```

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- ✓ ● The following will be logged to the console: "A" "C" "B"
- ○ The following will be logged to the console: "A" "B" "C"
- ○ The following will be logged to the console: "A" "B"
- ○ The following will be logged to the console: "C"

No Comments

QUESTION 20

✓ Correct Answer

Score 5

Numbers › Multiple Choice

Javascript | Javascript General Knowledge | Advanced

QUESTION DESCRIPTION

The following code will log 'false' to the console. **Why**?

```
console.log((0.1 + 0.2) === 0.3);
```

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- ○ The strong equality type check causes number comparisons to return false.
- ○ The value in the parenthesis are not evaluated by the time the console.log executes.
- ✓ ● Numbers are not stored as integers, but as 64bit IEEE 754 values.
- ○ The numbers are converted to strings before comparing.

No Comments

QUESTION 21

✓ Correct Answer

Score 5

Async/Await › Multiple Choice

Javascript | Javascript Async/Await | Advanced

QUESTION DESCRIPTION

Which of the following are benefits to using Async/Await?

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- ○ Highly compatible with older browsers
- ○ Highly compatible with older versions of JavaScript
- ○ Creates race conditions in your code
- ✓ ● Easy to enforce the order of execution after asynchronous functions
- ✓ ● Creates a clear separation of asynchronous tasks

No Comments

QUESTION 22

✗ Wrong Answer

Score 0

ES6 › Multiple Choice

Javascript | Javascript ES6 | Advanced

QUESTION DESCRIPTION

What will be logged to the console when the following code is executed?

```
const a = () => {5 + 5};

const b = a() + 5;

console.log(b);
```

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- ○ '[object Object]5'
- ✓ ○ NaN
- ● 15
- ○ This code will raise an exception.

No Comments

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