

Angular

Quiz and assignment 2

|  |  |
| --- | --- |
| Program Code |  |
| Issue/Revision | x/y |
| Effective date | May 2020 |

**Exam Day 2**

# Exam Instruction

* Total number of questions: 10
* Pass score: 5

# Exam Questions

1. What will the code below output to the console?

let number = 0;

console.log(number++);

console.log(++number);

console.log(number);

|  |  |
| --- | --- |
| ⭘ | A. 1 1 2 |
| ⭘ | B. 1 2 2 |
| ⭘ | **C. 0 2 2** |
| ⭘ | D. 0 1 2 |

1. What will the code below output to the console?

const foo = () => console.log("First");

const bar = () => setTimeout(() => console.log("Second"));

const baz = () => console.log("Third");

bar();

foo();

baz();

|  |  |
| --- | --- |
| ⭘ | A. First Second Third |
| ⭘ | **B. First Third Second** |
| ⭘ | C. Second First Third |
| ⭘ | D. Second Third First |

1. With which constructor can we successfully extend the Dog class?

class Dog {

constructor(name) {

this.name = name;

}

};

class Labrador extends Dog {

// 1

constructor(name, size) {

this.size = size;

}

// 2

constructor(name, size) {

super(name);

this.size = size;

}

// 3

constructor(size) {

super(name);

this.size = size;

}

// 4

constructor(name, size) {

this.name = name;

this.size = size;

}

};

|  |  |
| --- | --- |
| ⭘ | A. 1 |
| ⭘ | **B. 2** |
| ⭘ | C. 3 |
| ⭘ | D. 4 |

1. What will the code below output to the console?

function checkAge(age) {

if (age < 18) {

const message = "Sorry, you're too young.";

} else {

const message = "Yay! You're old enough!";

}

return message;

}

console.log(checkAge(21));

|  |  |
| --- | --- |
| ⭘ | A. "Sorry, you're too young." |
| ⭘ | B. "Yay! You're old enough |
| ⭘ | **C. ReferenceError** |
| ⭘ | D. undefined |

1. What will the code below output to the console?

function Cat(fullName) {

this.fullName = fullName;

this.sayHi = function() {

return function(){console.log(this.fullName)};

}

}

function Mouse(fullName) {

this.fullName = fullName;

this.sayHi = function() {

return () => {console.log(this.fullName)};

}

}

const tom = new Cat('Tom');

const jerry = new Mouse('Jerry');

tom.sayHi()();

jerry.sayHi()();

|  |  |
| --- | --- |
| ⭘ | A. Tom Jerry |
| ⭘ | **B. undefined Jerry** |
| ⭘ | C. undefined undefined |

1. What will the code below output to the console? (Single selection)

// module.js

export default () => 'Hello world';

export const name = 'Lydia';

// index.js

import \* as data from './module';

console.log(data);

|  |  |
| --- | --- |
| ⭘ | A. { default: function default(), name: "Lydia" } |
| ⭘ | B. { default: function default() } |
| ⭘ | **C. { default: "Hello world", name: "Lydia" }** |

1. What will the code below output to the console? (Single selection)

class Person {

constructor(name) {

this.name = name;

}

}

const member = new Person('John');

console.log(typeof member);

|  |  |
| --- | --- |
| ⭘ | A. "class" |
| ⭘ | B. "function" |
| ⭘ | **C. "object"** |
| ⭘ | D. "string" |

1. What will the code below output to the console?

const myPromise = () => Promise.resolve('I have resolved!');

function firstFunction() {

myPromise().then(res => console.log(res));

console.log('second');

}

async function secondFunction() {

console.log(await myPromise());

console.log('second');

}

firstFunction();

secondFunction();

|  |  |
| --- | --- |
| ⭘ | A. I have resolved!, second I have resolved!, second |
| ⭘ | B. second, I have resolved! second, I have resolved! |
| ⭘ | C. I have resolved!, second second, I have resolved! |
| ⭘ | **D. second, I have resolved! I have resolved!, second** |

1. What is the difference between the map and forEach Array methods?

|  |  |
| --- | --- |
| ⭘ | A. forEach loops over the array forwards while map loops backwards |
| ⭘ | B. map loops over the array forwards while forEach loops backwards |
| ⭘ | C. forEach returns a new array while map does not |
| ⭘ | **D. map returns a new array while forEach does not** |

1. Function A declares variable X and returns function B. Can function B access variable X?

|  |  |
| --- | --- |
| ⭘ | A. Yes, because of the scope chain |
| ⭘ | B. No, because of the prototype chain |
| ⭘ | C. Yes, because of closures |
| ⭘ | **D. No, because functions are not suppose to access outer variables** |