MEAN Stack Interview Questions & Answers

Explain the MEAN Stack Architecture

The primary function of the different components of Mean Stack Architecture are as follows:

- **Angular JS: Accepts requests and displays results to the end user**
- **NodeJS: Handles client and server requests**
- **MongoDB: Stores and retrieves data.**
- **Express JS: Make requests to the Database and return a response**

Your passion and zeal will shine through in these Salesforce lightning MCQ multiple-choice questions; the more each of you takes them, the better your results will be.

1. What is Mean Stack?

JavaScript framework
[✓ ***A collection of JavaScript technologies for creating web applications***]
Back-end technology
Front-end technology

2. Which one is a significant advantage of Mean Stack?

[✓ ***It allows developers to develop front-end and back-end components separately.***]
Uses a complex format for data exchange
Suitable for all kinds of applications
Requires less setup

3. What does Mean Stack use the data exchange format?

XML
[✓ ***JSON***]
YAML
CSV

4. For whom can use Mean Stack be beneficial?

Who are new to JavaScript
[✓ ***Those with a good understanding of JavaScript***]
I prefer statically typed languages.
multiple programming languages for web applications

5. Which web application framework is discussed?

Flask
[✓ ***Express***]
Django
Ruby on Rails

6. What is the primary purpose of Express?

To reduce boilerplate code and build standard web application features ✓
Create complex databases
Provide a front-end user interface.
Write complex algorithms

7. Whichcombines static views and dynamic data to generate dynamic content for the user?

Routing
Templating ✓
Middleware
Authentication

8. What is the definition of middleware in Express?

Handle user authentication
Send specific headers
Manipulate the application's behaviour based on the request.
All of the above ✓

9. What remains the primary reason Express is an excellent option for front-end developers?

Complex algorithms
Back-end development ✓
Robust database management system
Complex front-end user interface

10. Which one is MongoDB?

Server-side web application framework
A document-based no-SQL database system ✓
Front-end user interface
Complex algorithm

What is REPL in Node.Js?

Ans: REPL stands for Reading Eval Print Loop. It also evaluates datasets and displays the results. It is essential to understand that REPL replicates the environment of a Linux shell. REPL is related to the successful completion of the tasks listed below.

- **Print:** Print links with printing the results derived from the command.
- **Read** The read command links with reading the inputs provided by the user. It stores the structure inside the function.
- **Loop:** Loop would continue to twist the command until the user uses Ctrl+C.

Describe Express.js routing?

Ans: Frameworks utilize routing to determine how a URL/endpoint responds to/is handled by the server. Express is a good tool for managing application routing. The following is some basic Express routing code.

```
var express = require('express')

var app = express()

// respond with "mean stack interview questions" when a GET request is made to the h
omepage

app.get('/', function (req, res) {
  res.send('Mean stack interview questions')
});
```

What exactly is Dependency Injection?

Ans: Dependency injection is simple to design loosely connected components, which often implies that components consume interface-defined functionality without knowing which implementation classes are utilized.

What is DATA modeling?
Ans: Data modeling assists in visually representing data while simultaneously imposing corporate standards, regulatory compliances, and government policies on the data. It ensures consistency in naming standards, default values, semantics, data security, and data quality.

Differentiate between Node.js, AJAX, and jQuery.
Ans: Node.js, AJAX, and jQuery are all advanced JavaScript implementations. Node.js is a server-side platform for creating client-server applications. In contrast, AJAX, Asynchronous Javascript, and XML) are client-side scripting techniques used to render a page's contents without refreshing it. AJAX is used primarily to display dynamic pages. jQuery is a JavaScript package that works with AJAX, DOM traversal, and looping. It includes a variety of useful utilities to work with JavaScript programming.

What is a CSS Grid System?
Ans: A grid system in CSS is a structure used for storing content vertically and horizontally in a uniform and understandable manner. Rows and columns are the two main components of grid systems. Simple, Pure, Flexbox, Bootstrap, and Foundation are some of the most popular grid systems.

What is CORS, and how do you enable it?
Ans: CORS (cross-origin resource sharing) is a protocol that manages cross-origin requests. CORS enables servers to determine who (i.e., which sources) can access the server's assets, among other things.
The Access-Control-Allow-Origin HTTP header specifies which foreign origins are permitted to access the content of your domain's pages via scripts using methods such as XMLHttpRequest.

n Javascript, describe WeakSet.
Ans: A Set in javascript is a collection of unique and ordered elements. WeakSet is the collection of unique and ordered elements with the following differences:
Weakset holds objects, no other types.

A weakly referred item is contained within the weakset, Which means that if the item within the weakset lacks , it will be junk collected.
WeakSet offers only three methods: add(), remove(), and has().

```
const data = new Set([1,2,3,4]);
console.log(data);// Outputs Set {1,2,3,4}

const data2 = new WeakSet([3, 4, 5]); //Throws an error

        let obj1 = {message:"Hello Ninjas"};

const data3= new WeakSet([obj1]);
console.log(data3.has(obj1)); // true
```

In Javascript, explain WeakMap.
Ans: The map is used in javascript to store key-value pairs. The key-value pairs might be either primitive or non-primitive. WeakMap is similar to Map, but with the following important differences:

- In weakmap, the keys and values should always be objects.
- If no references exist, it will be garbage collected.

```
const data1= new Map();
data1.set('Value', 10);


const data2 = new WeakMap();
data2.set('Value', 22.3); // Throws an error


let obj = {name:"Ninjas"};
const data3 = new WeakMap();
data3.set(obj, {age:13});
```

What is Routing Guard in Angular?
Ans: Angular’s route guards are interfaces that can tell the router whether it should allow navigation to a requested route or not. Routing Guard makes this decision by looking for a true or false return value from a class that *implements* the given guard interface.
There are five different types of guards called in a particular sequence. The router’s behavior is modified differently depending on which guard is used. The guards are:

- CanActivate
- CanActivateChild
-
- CanDeactivate
- CanLoad
- Resolve

Consider the below example:

```
import { Injectable } from '@angular/core';
import { Router, CanActivate } from '@angular/router';
import { AuthService } from './auth.service';

@Injectable()
export class coding ninjas implements CanActivateclass {
  constructor(public auth: AuthService, public router: Router) {}
  canActivateclass(): boolean {
    if (!this.auth.isAuthenticated()) {
      this.router.navigate(['login']);
      return false;
    }
    return true;
  }
}
```

29. Differentiate between “ == “ and “ === “ operators.
Ans: They are both comparison operators. The difference between the two is that "==" compares values whereas " === " compares both values and types.
Example:

```
var x = 3;
var y = "3";

(x == y)  // Returns true as the value in x and y are the same
(x === y) // Returns false as the type of x is "number" and the type of y is "string".
```

30. What is NaN in JavaScript?
Ans: NaN represents the “Not-a-Number” value. It indicates a value which is not a legal number.
type of NaN returns a Number.
To verify if the number is NaN, we will use the isNaN() function,
Note- isNaN() function changes the given value to a Number type, and then equates it to NaN.

```
isNaN("Hello ninjas")  // Returns true
isNaN(1234)   // Returns false
isNaN('1')  // Returns false, as '1' will convert to a Number type which results in 0 ( a number)
isNaN(true) // Returns false, as true will convert to Number type results in 1 ( a number)
isNaN(true) // Returns true
isNaN(undefined) // Returns true
```

Do you think that TypeScript supports all object-oriented principles?

Yes, TypeScript supports all the object-oriented principles. There are four main principles of Object Oriented Programming:

- Inheritance,
- Encapsulation,
- Polymorphism and

- Abstraction

TypeScript can implement all four object-oriented programming principles with a cleaner and smaller syntax.

Define Cross-site Scripting (XSS).

Cross-site Scripting (XSS) is a client-side code injection attack wherein the malicious scripts are executed in a web browser by including malicious code in a legitimate web page or web application. It can also occur when an individual clicks on untrusted links that can pass cookies and other sensitive information to attackers.

The attack happens when you visit a webpage or a web app that executes malicious code. Hence, the webpage or web app becomes a vehicle to deliver malicious scripts to the browser of a user.

The most commonly used vehicles for cross-site scripting attacks are forums, message boards, and even web pages that encourage users to comment.

What does signify "non-obstructing" in node.js?

In node.js "non-blocking" implies that its IO is non-blocking. Hub utilizes "libuv" to deal with its IO in a stage rationalist manner. On windows, it utilizes finish ports for UNIX it utilizes epoll or kqueue, and so on. Along these lines, it makes a non-blocking demand and upon demand, it lines it inside the occasion circle which calls the JavaScript 'callback' on the fundamental JavaScript string.

What are the two sorts of API works in Node.js?

The two kinds of API that work in Node.js are

- Asynchronous, non-blocking capacities
- Synchronous, blocking capacitie

```
});
```

A) `Error reading file: ENOENT: no such file or directory, open 'nonexistent.txt'`
B) `undefined`
C) `Error`
D) `ReferenceError`

2. What is the purpose of the `process.env.NODE_ENV` variable in Node.js?

   A) To manage environment-specific configurations.
   B) To declare global variables.
   C) To store user session data.
   D) To control HTTP headers.

3. How can you handle synchronous operations in Node.js?

   A) Using callbacks.
   B) Using `async/await`.
   C) Using `setTimeout()`.
   D) All of the above.

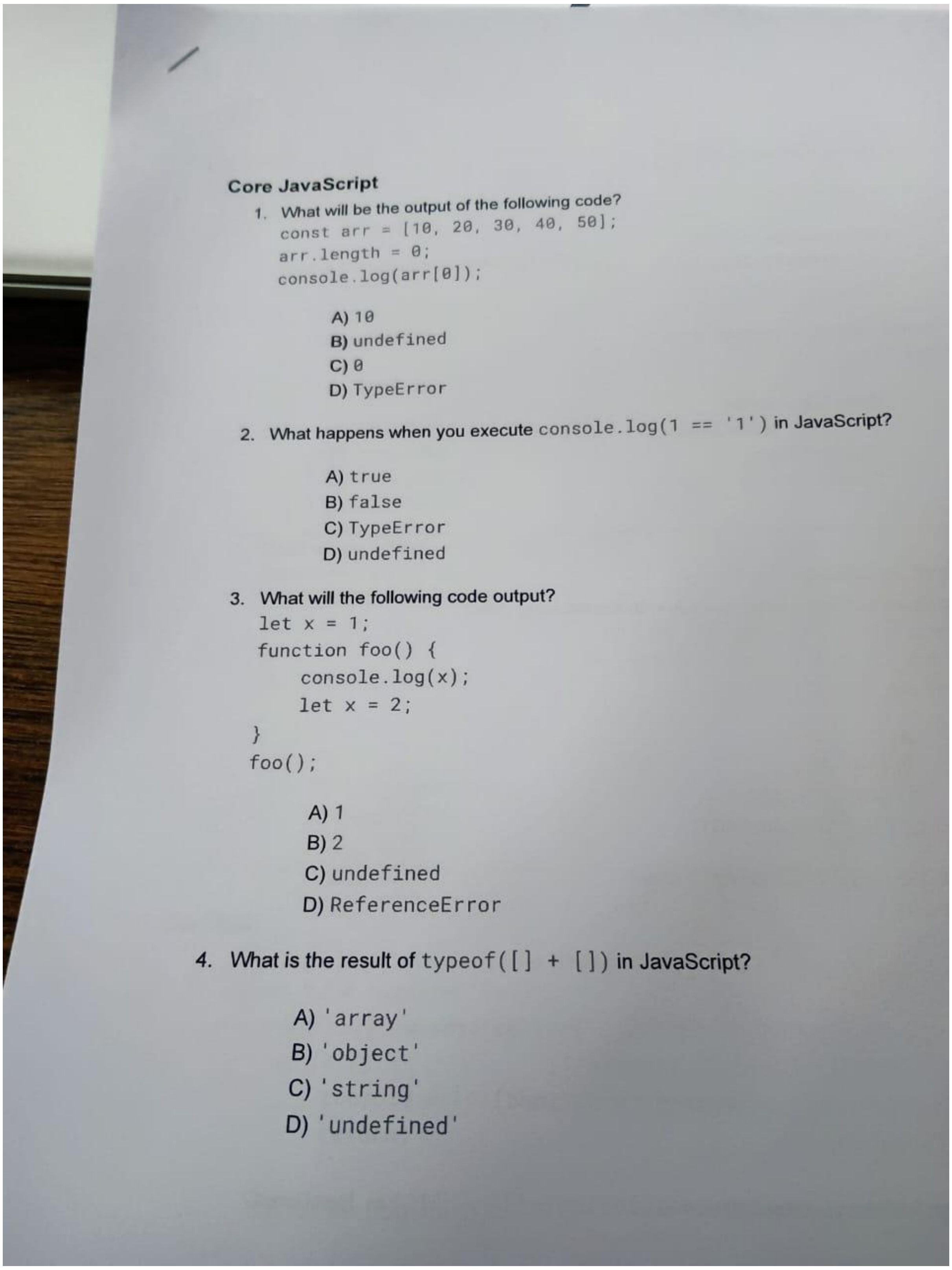
4. What is the difference between `setImmediate()` and `process.nextTick()` in Node.js?

   A) `setImmediate()` is used for scheduling callbacks, whereas `process.nextTick()` executes callbacks immediately after the current operation.
   B) `setImmediate()` executes callbacks immediately, whereas `process.nextTick()` schedules callbacks.
   C) Both are used for scheduling callbacks with the same behavior.
   D) `setImmediate()` is deprecated in favor of `process.nextTick()`.

5. How can you import modules in Node.js?

   A) Using `require('./module')`.
   B) Using `import module from './module'`.
   C) Using `use('./module')`.
   D) Using `load('./module')`.

ReactJS



Core JavaScript

1. What will be the output of the following code?

```
const arr = [10, 20, 30, 40, 50];
arr.length = 0;
console.log(arr[0]);
```

   A) `10`
   B) `undefined`
   C) `0`
   D) `TypeError`

2. What happens when you execute `console.log(1 == '1')` in JavaScript?

   A) `true`
   B) `false`
   C) `TypeError`
   D) `undefined`

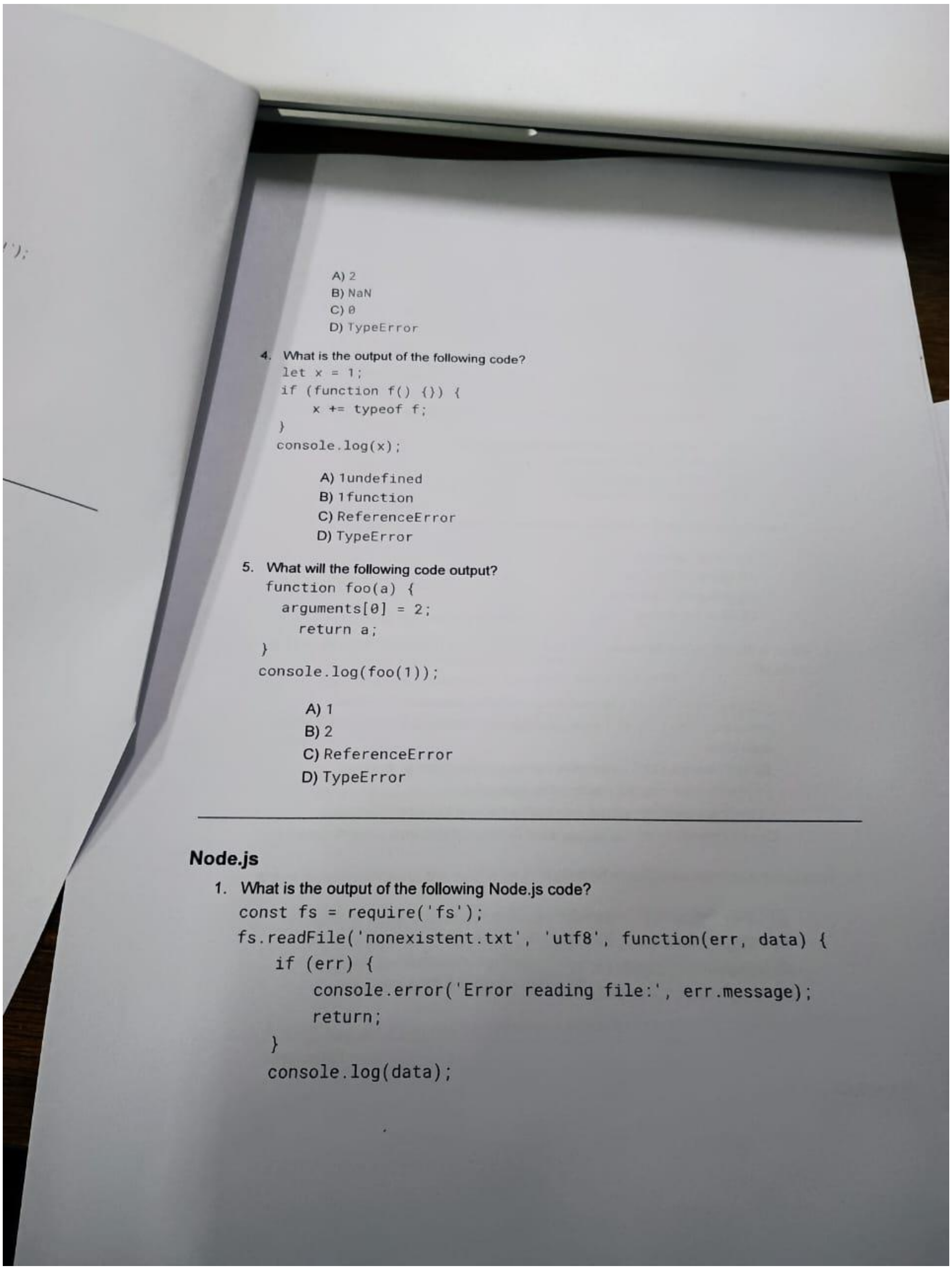
3. What will the following code output?

```
let x = 1;
function foo() {
    console.log(x);
    let x = 2;
}
foo();
```

   A) 1
   B) 2
   C) `undefined`
   D) `ReferenceError`

4. What is the result of `typeof([] + [])` in JavaScript?

   A) `'array'`
   B) `'object'`
   C) `'string'`
   D) `'undefined'`



A) 2
B) NaN
C) 0
D) TypeError

4. What is the output of the following code?

```
let x = 1;
if (function f() {}) {
    x += typeof f;
}
console.log(x);
```

A) 1undefined
B) 1function
C) ReferenceError
D) TypeError

5. What will the following code output?

```
function foo(a) {
  arguments[0] = 2;
    return a;
}
console.log(foo(1));
```

A) 1
B) 2
C) ReferenceError
D) TypeError

Node.js

1. What is the output of the following Node.js code?

```
const fs = require('fs');
fs.readFile('nonexistent.txt', 'utf8', function(err, data) {
    if (err) {
        console.error('Error reading file:', err.message);
        return;
    }
    console.log(data);
```

5. What is the output of the following code?

```
const person = {
 name: 'John',
    greet: function() {
        console.log(`Hello, ${this.name}!`);
    }
};
const greet = person.greet;
greet();
```

A) `Hello, John!`
B) `Hello, undefined!`
C) `TypeError`
D) `SyntaxError`

Advanced JavaScript

1. What will be logged to the console with the following code?

```
const arr = [10, 20, 30, 40, 50];
const slicedArr = arr.slice(2, 4);
console.log(slicedArr);
```

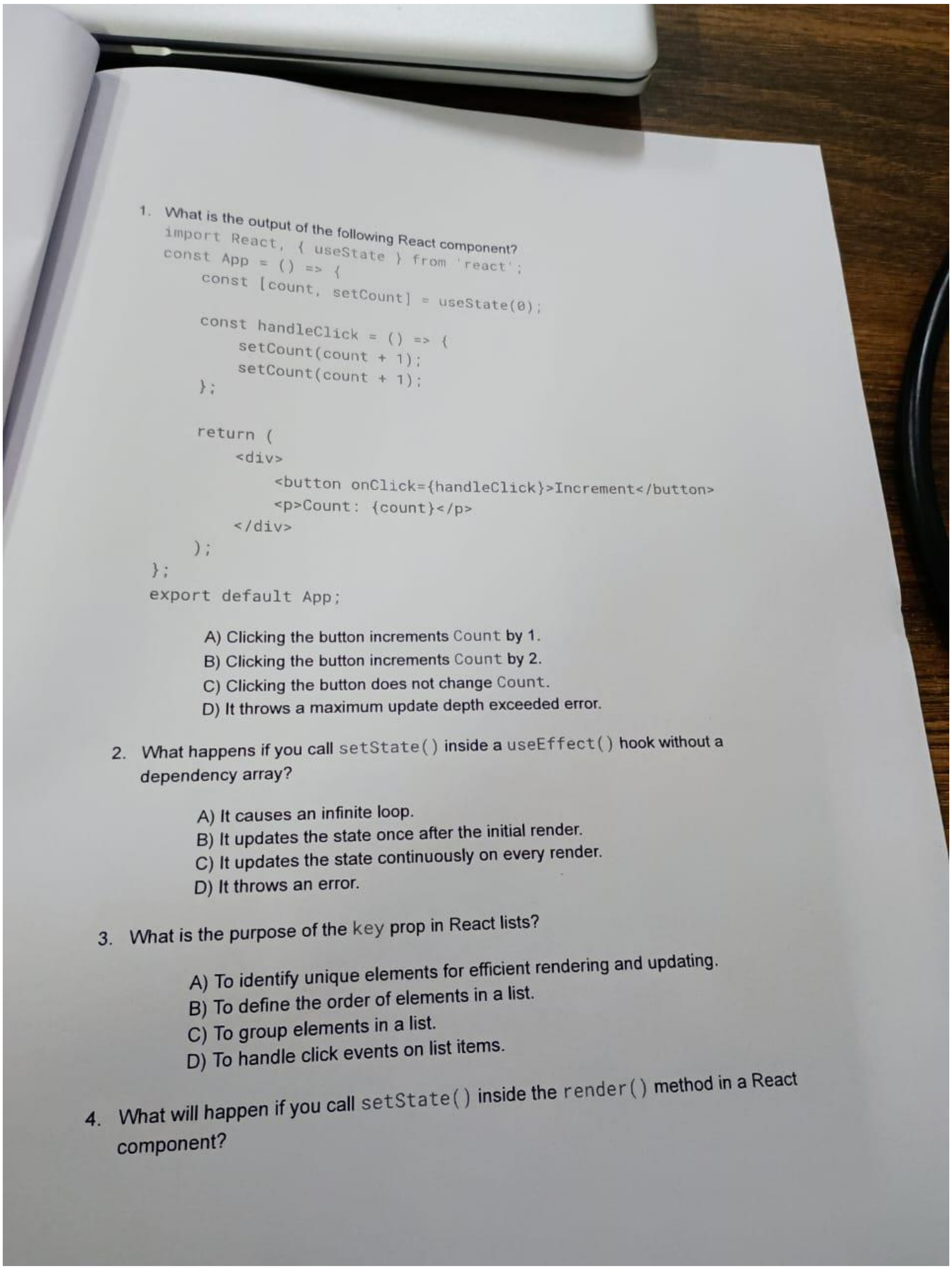
A) `[30, 40]`
B) `[20, 30]`
C) `[30, 40, 50]`
D) `[10, 20, 30, 40]`

2. What is the output of the following code?

```
const obj = { 1: 'one', 2: 'two', 3: 'three' };
const arr = [...obj];
console.log(arr);
```

A) `[{ 1: 'one' }, { 2: 'two' }, { 3: 'three' }]`
B) `[['one'], ['two'], ['three']]`
C) `[undefined, undefined, undefined]`
D) `TypeError`

3. What happens when you execute `console.log(+'1' - '1')` in JavaScript?



1. What is the output of the following React component?

```
import React, { useState } from 'react';
const App = () => {
    const [count, setCount] = useState(0);

    const handleClick = () => {
        setCount(count + 1);
        setCount(count + 1);
    };

    return (
        <div>
            <button onClick={handleClick}>Increment</button>
            <p>Count: {count}</p>
        </div>
    );
};
export default App;
```

A) Clicking the button increments `Count` by 1.
B) Clicking the button increments `Count` by 2.
C) Clicking the button does not change `Count`.
D) It throws a maximum update depth exceeded error.

2. What happens if you call `setState()` inside a `useEffect()` hook without a dependency array?

A) It causes an infinite loop.
B) It updates the state once after the initial render.
C) It updates the state continuously on every render.
D) It throws an error.

3. What is the purpose of the `key` prop in React lists?

A) To identify unique elements for efficient rendering and updating.
B) To define the order of elements in a list.
C) To group elements in a list.
D) To handle click events on list items.

4. What will happen if you call `setState()` inside the `render()` method in a React component?