JS ASSESSMENT - 01

For All Candidates:

Refactor below ES5 code using ES6 Features:

```
// 1.
var numbers = [1, 2, 3, 4, 5];
var squares = [];
for (var i = 0; i < numbers.length; i++) {
  squares.push(numbers[i] * numbers[i]);
}

// 2.
function Person(name, age) {
  this.name = name;
  this.age = age;
}
Person.prototype.greet = function() {
  return 'Hello, my name is ' + this.name + ' and I am ' + this.age + ' years old.';
};
var john = new Person('John', 30);

// 3.
try {
  // Some code that might throw an error
} catch (e) {
  console.error('An error occurred: ' + e.message);
}

// 4.
function fetchData(callback) {
  setTimeout(function() {
    callback('Data fetched');
  }, 1000);
}

// 5.
var module = (function() {
  var privateVar = 10;
  function privateFunction() {
    return privateVar;
  }
  return {
    getPrivateVar: privateFunction
  };
})();
```

```
// 6.
document.getElementById('myButton').onclick = function(event) {
  console.log('Button clicked');
};

// 7.
var data = [
  { name: 'John', age: 30 },
  { name: 'Alice', age: 25 },
  { name: 'Bob', age: 35 }
];
var filteredData = data.filter(function(person) {
  return person.age > 30;
});

// 8.
function doubleAndFilter(numbers) {
  var doubled = numbers.map(function(num) {
    return num * 2;
  });
  return doubled.filter(function(num) {
    return num % 3 === 0;
  });
}

// 9.
function fetchData() {
  return new Promise(function(resolve, reject) {
    // Some asynchronous operation
    if (/* operation successful */) {
      resolve('Data fetched');
    } else {
      reject('Error fetching data');
    }
  });
}

// 10.
function Animal(name) {
  this.name = name;
}
Animal.prototype.sayName = function() {
  console.log('My name is ' + this.name);
};
var dog = new Animal('Buddy');
```

For All Candidates:

Problem 1: Write a function that prints the numbers from 1 to n. But for multiples of three, print "Fizz" instead of the number, and for the multiples of five, print "Buzz". For numbers that are multiples of both three and five, print "FizzBuzz".

Problem 2: Write a function to sort an array of numbers in ascending order without using the built-in sort() method.

For Each Candidate Separately:

Ahsan Khan
Imagine you're convincing a colleague who primarily works with server-side languages to learn JavaScript. How would you explain the significance of JavaScript in modern web development and its benefits, especially in terms of client-side interactivity and versatility across different platforms?

Ankita Agarwal
You're leading a team discussion on the differences between ES5 and ES6 syntax. How would you illustrate with practical examples the advantages of ES6 features such as arrow functions, template literals, and destructuring over their ES5 counterparts?

Devesh Singh
Suppose you're building a calculator application in JavaScript. How would you utilize different types of operators (arithmetic, assignment, comparison, logical) to implement basic arithmetic operations and ensure accurate calculation results?

Rashmi Nainwal
You're tasked with refactoring a legacy codebase written in ES5 to utilize ES6 features. Can you provide specific examples of how you would use let/const declarations, template literals, destructuring, default parameters, arrow functions, and spread/rest operators to improve code readability and maintainability?

Dhirendra Singh
You're developing a form validation function in JavaScript. How would you leverage short-circuit evaluation to efficiently check for empty or invalid input fields and provide appropriate feedback to the user without unnecessary computation?

Gautam Mahto
Imagine you're implementing a user authentication system in a web application. How would you use conditional statements (if/else, switch) to handle different authentication scenarios, such as valid credentials, expired sessions, or unauthorized access attempts?

Neeraj Sharma
You're working on a data retrieval function that may return null or undefined values in certain cases. How would you use the nullish coalescing operator (??) to provide fallback values or handle null/undefined cases gracefully without causing errors?

Sitara Hussain
Suppose you're accessing nested properties of an object received from an API response. How would you use optional chaining (?.) to safely navigate through the object's structure and handle cases where certain properties may be null or undefined?

Soumya Ranjan
You're developing a data processing module that deals with various data types, including strings, numbers, arrays, and objects. How would you ensure proper type checking, conversion, and manipulation to maintain data integrity and consistency throughout the application?

Ubaid Qureshi
Imagine you're implementing a feature that involves iterating over an array of user objects and performing different operations such as filtering, mapping, and reducing. How would you use array iteration methods like forEach, filter, map, reduce, from, and some to achieve the desired functionality efficiently and effectively?