

## A. Answer the following:

### 1. JavaScript vs Typescript.

**Answer:** JavaScript and TypeScript are both widely used in web development, but they have fundamental differences that cater to different programming needs.

- **Typing System:** JavaScript is a dynamically typed language, which means that variables can hold values of any type and can change types at runtime. This flexibility can lead to runtime errors if types are not managed carefully. In contrast, TypeScript is a statically typed superset of JavaScript. It introduces optional static typing, allowing developers to define variable types explicitly. This feature helps catch errors during compilation rather than at runtime.
- **Error Handling:** In JavaScript, type checking occurs at runtime, potentially resulting in unexpected behavior if incorrect types are used. TypeScript's compile-time error checking reduces the likelihood of bugs in production by catching type-related errors early in the development process.
- **Features:** TypeScript includes advanced features such as interfaces, generics, and decorators that enhance code organization and maintainability—features that are not available in JavaScript.

### 2. How to use Typescript?

**Answer:** Using TypeScript involves several straightforward steps:

1. **Installation:** To begin, install TypeScript globally using npm (Node Package Manager) with the following command:

```
npm install -g typescript
```

2. **Creating a TypeScript File:** Next, create a new file with a .ts extension (for example, app.ts) where you will write your TypeScript code.

3. **Writing Code:** In your TypeScript file, write your code with type annotations. For instance:

```
let message: string = "Hello, TypeScript!";
```

4. **Compiling Code:** To convert your TypeScript code into JavaScript, compile the file using the command:

**tsc app.ts**

5. **Running the Code:** You can run the compiled JavaScript file using Node.js or include it in an HTML file for execution in a web browser.

6. **Using Type Definitions:** To enhance type safety and provide IntelliSense support in your Integrated Development Environment (IDE), leverage type definitions from DefinitelyTyped for third-party libraries.

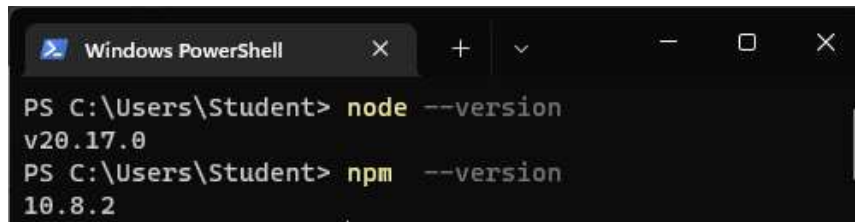
### 3. TypeScript Features?

**Answer:** TypeScript offers several notable features that significantly improve the development experience:

- **Static Type Checking:** This feature enables early detection of type-related errors before the code is executed, enhancing reliability.
- **Interfaces:** TypeScript allows developers to define interfaces that specify the structure of objects. This ensures that objects adhere to defined formats, improving code consistency.
- **Generics:** Generics enable developers to create reusable components that work with various data types while maintaining type safety.
- **Decorators:** Decorators provide a way to modify classes and methods at design time, allowing for more flexible and organized code design.
- **Enhanced Tooling Support:** Due to its static typing, TypeScript offers better autocompletion and refactoring capabilities in IDEs, making it easier for developers to write and maintain their code.

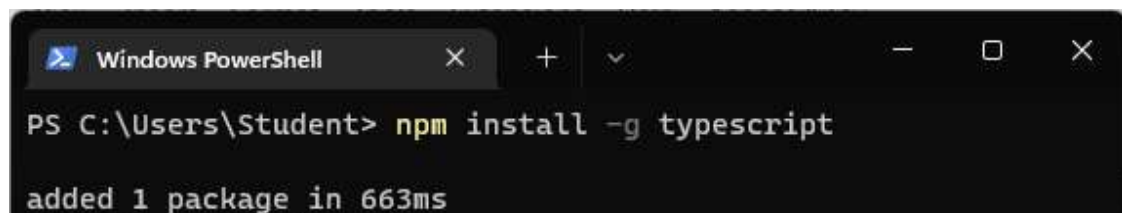
## B. Attach Screenshot

- **Typescript Installation**



```
Windows PowerShell
PS C:\Users\Student> node --version
v20.17.0
PS C:\Users\Student> npm --version
10.8.2
```

*Check the Node and npm versions.*



```
Windows PowerShell
PS C:\Users\Student> npm install -g typescript

added 1 package in 663ms
```

*Use the command `npm install -g typescript` to install TypeScript globally.*

- **Typescript Hello World**



```
helloworld.ts
1 console.log("Hello World !")
```

Above is the TypeScript code for the `helloworld.ts` file that outputs 'Hello World!'



```
Windows PowerShell
PS C:\VishalRMahajan_TEITA4> tsc helloworld.ts
PS C:\VishalRMahajan_TEITA4> node helloworld
Hello World !
PS C:\VishalRMahajan_TEITA4> |
```

*The screenshot above shows the output of the 'Hello, World!' code.*

**Q. Write a program that includes the following components:**

1. Environment setup
2. Variable declaration with type checking
3. Control structures: if-else, switch-case, and loops
4. Function implementation
5. Use of the ternary operator

**Answer:**

```
PS C:\VishalRMahajan_Web> npm init -y
Wrote to C:\VishalRMahajan_Web\package.json:

{
  "name": "vishalrmahajan_web",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}
```

*Initializing a Node.js project.*

```
PS C:\VishalRMahajan_Web> npm install typescript --save-dev
added 1 package, and audited 2 packages in 1s
found 0 vulnerabilities
PS C:\VishalRMahajan_Web> npx tsc --init
Created a new tsconfig.json with:

target: es2016
module: commonjs
strict: true
esModuleInterop: true
skipLibCheck: true
forceConsistentCasingInFileNames: true

You can learn more at https://aka.ms/tsconfig
```

*Installs TypeScript and creates configuration files.*

```
PS C:\VishalRMahajan_Web> npm install prompt-sync
added 3 packages, and audited 5 packages in 1s
found 0 vulnerabilities
PS C:\VishalRMahajan_Web> npm install @types/node --save-dev
added 2 packages, and audited 7 packages in 874ms
found 0 vulnerabilities
```

## Code:

```
const promptSync = require("prompt-sync")();
// Variable Declaration and Type Checking
type Product = {
  id: number;
  name: string;
  price: number;
  quantity: number;
};

let cart: Product[] = [];
let isActive: boolean = true;

// Function
function addProductToCart(product: Product): void {
  let existingProduct = cart.find((item) => item.id === product.id);

  //if-else
  if (existingProduct) {
    existingProduct.quantity += product.quantity;
  } else {
    cart.push(product);
  }
}

// Switch-Case
function cartAction(action: string): string {
  switch (action) {
    case "view":
      if (cart.length === 0) {
        return "Cart is empty.";
      }
  }
}
```

```

    let details = cart.map(
      (item) => `${item.name}: ${item.quantity} x ₹${item.price}`
    );
    return `Products in cart:\n${details.join("\n")}`;

    case "checkout":
      if (cart.length === 0) {
        return "Cannot checkout, cart is empty.";
      }
      isCartActive = false;
      return "Checkout successful!";

    default:
      return "Invalid action.";
  }
}

```

// Loop

```

while (isCartActive) {
  console.log("\nShopping Cart Menu:");
  console.log("1. Add Product");
  console.log("2. View Cart");
  console.log("3. Checkout");
  console.log("4. Exit");

  const choice = promptSync("Enter your choice (1-4): ") || "";

  switch (choice) {
    case "1":
      const id = Number(promptSync("Enter product ID: ") || 0);
      const name = promptSync("Enter product name: ") || "";
      const price = Number(promptSync("Enter product price: ") || 0);

```

```
    const quantity = Number(promptSync("Enter product quantity: ")
|| 0);
    addProductToCart({ id, name, price, quantity });
    console.log(`Added ${name} to the cart.`);
    break;

case "2":
    console.log(cartAction("view"));
    break;

case "3":
    console.log(cartAction("checkout"));
    break;

case "4":
    isCartActive = false;
    console.log("Exiting. Goodbye!");
    break;

default:
    console.log("Invalid choice. Please try again.");
}

// Ternary Operator
const cartStatus = isCartActive ? "Cart is active." : "Cart is
inactive.";
console.log(cartStatus);
}
```

## Output:

```
PS C:\VishalRMahajan_Web> npx tsc .\shoppingcart.ts
PS C:\VishalRMahajan_Web> node .\shoppingcart.js
```

```
Shopping Cart Menu:
1. Add Product
2. View Cart
3. Checkout
4. Exit
Enter your choice (1-4): 1
Enter product ID: 1
Enter product name: Laptop
Enter product price: 60000
Enter product quantity: 1
Added Laptop to the cart.
Cart is active.
```

*Running the program and adding product to cart*

```
Shopping Cart Menu:
1. Add Product
2. View Cart
3. Checkout
4. Exit
Enter your choice (1-4): 2
Products in cart:
Laptop: 1 x ₹60000
Cart is active.
```

*Viewing the content in the cart*

```
Shopping Cart Menu:
1. Add Product
2. View Cart
3. Checkout
4. Exit
Enter your choice (1-4): 3
Checkout successful!
Cart is inactive.
PS C:\VishalRMahajan_Web> |
```

*Checking out from the cart*