Question 1: What is error handling in JavaScript? Explain the try, catch, and finally blocks with an example.

**✅ Error Handling in JavaScript**

**Error handling** in JavaScript allows you to **gracefully handle runtime errors** (like dividing by zero or accessing undefined variables) so your program doesn’t crash.

JavaScript provides the try, catch, and finally blocks to manage errors.

**🔹 Syntax**

*try {*

*// Code that might throw an error*

*} catch (error) {*

*// Code to handle the error*

*} finally {*

*// (Optional) Code that runs no matter what*

*}*

**🔸 Explanation of Blocks:**

| **Block** | **Description** |
| --- | --- |
| try | Contains code that **might throw an error** |
| catch | Runs **if an error occurs** in the try block; handles the error |
| finally | (Optional) Executes **after try and catch**, regardless of success/fail |

**🔹 Example**

*try {*

*let num = 10;*

*let result = num / 0;*

*if (!isFinite(result)) {*

*throw new Error("Cannot divide by zero!");*

*}*

*console.log(result);*

*} catch (error) {*

*console.error("Error caught:", error.message);*

*} finally {*

*console.log("This block always runs.");*

*}*

**🔍 Output:**

Error caught: Cannot divide by zero!

This block always runs.

Question 2: Why is error handling important in JavaScript applications?

**✅ Why is Error Handling Important in JavaScript Applications?**

Error handling is **crucial** in JavaScript applications to ensure that your program continues to run smoothly, even when something goes wrong.

**🔹 Reasons Why Error Handling Is Important:**

|  | **Reason** | **Explanation** |
| --- | --- | --- |
| 1 | **Prevents App Crashes** | Without handling errors, a small mistake can stop the entire app. |
| 2 | **Improves User Experience** | Shows friendly error messages instead of technical ones or blank screens. |
| 3 | **Debugging and Logging** | Helps developers understand what went wrong using error logs or messages. |
| 4 | **Maintains Code Flow** | Allows execution to continue even if part of the code fails. |
| 5 | **Security** | Prevents exposing sensitive information through raw error messages. |
| 6 | **Cleaner and Reliable Code** | Separates normal code from error-handling logic, making code neater. |

**🔸 Example Without Error Handling**

*let x = y + 1; // y is not defined → causes app to crash*

*console.log("This line will never run");*

**🔸 Example With Error Handling**

*try {*

*let x = y + 1;*

*} catch (err) {*

*console.error("An error occurred:", err.message);*

*}*

*console.log("App continues running...");*