1. 3 Error Handling

Exceptions are a crucial aspect of handling runtime errors in JavaScript. By understanding and implementing proper error handling, you can build more resilient applications.

1. Error Types

- **SyntaxError**: Occurs when there is an invalid syntax in the code (e.g., a missing parenthesis or bracket).

  ```
  const x = (5 + 3;
  // SyntaxError: Unexpected token
  ```

- **ReferenceError**: This happens when a variable or function is referenced that doesn't exist or is out of scope.

  ```
  console.log(nonExistentVariable);
  // ReferenceError: nonExistentVariable is not defined
  ```

- **TypeError**: Occurs when an operation or method is applied to an incompatible type.

  ```
  let a = "hello";
  let b = a * 2;
  // TypeError: Cannot perform arithmetic on a string
  ```

2. Manually Throw Exceptions

You can manually throw an exception using the `throw` statement. You can throw anything (strings, objects, or any value), but typically, you throw an instance of an `Error` object or a subclass of it.

```
throw new Error('Something went wrong');
```

This can then be caught using a `try-catch` block.

3. Handling Exceptions with try-catch-finally

▼ JavaScript try...catch Statement

```
try {
    // body of try
}
catch(error) {
    // body of catch
}
```

The main code is inside the `try` block. While executing the `try` block, if any error occurs, it goes to the `catch` block. The `catch` block handles the errors as per the catch statements.

If no error occurs, the code inside the `try` block is executed and the `catch` block is skipped.

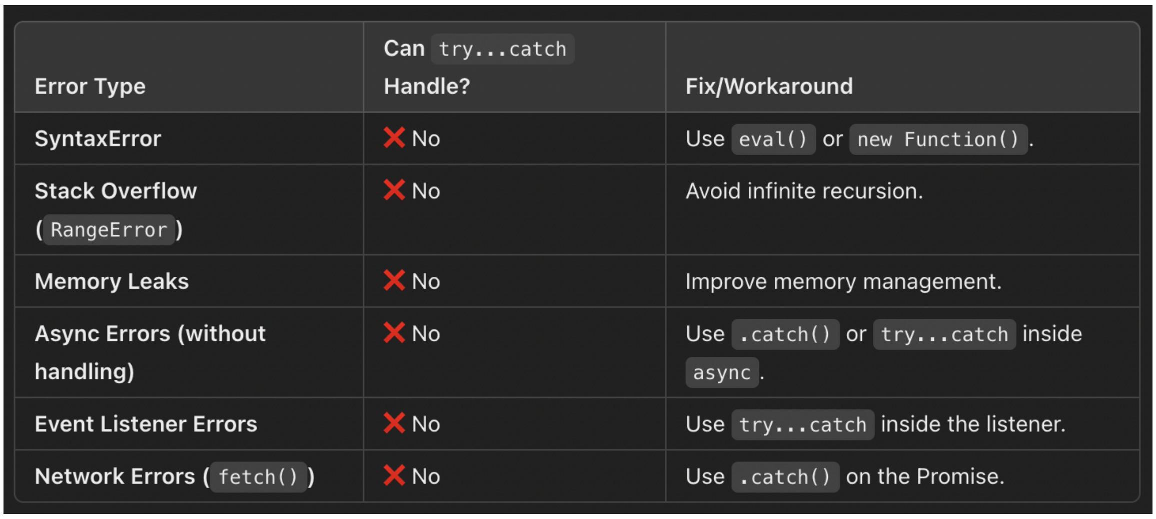
▼ JavaScript try...catch...finally Statement

You can also use the `try...catch...finally` statement to handle exceptions. The `finally` block executes both when the code runs successfully or if an error occurs.

The syntax of `try...catch...finally` block is:

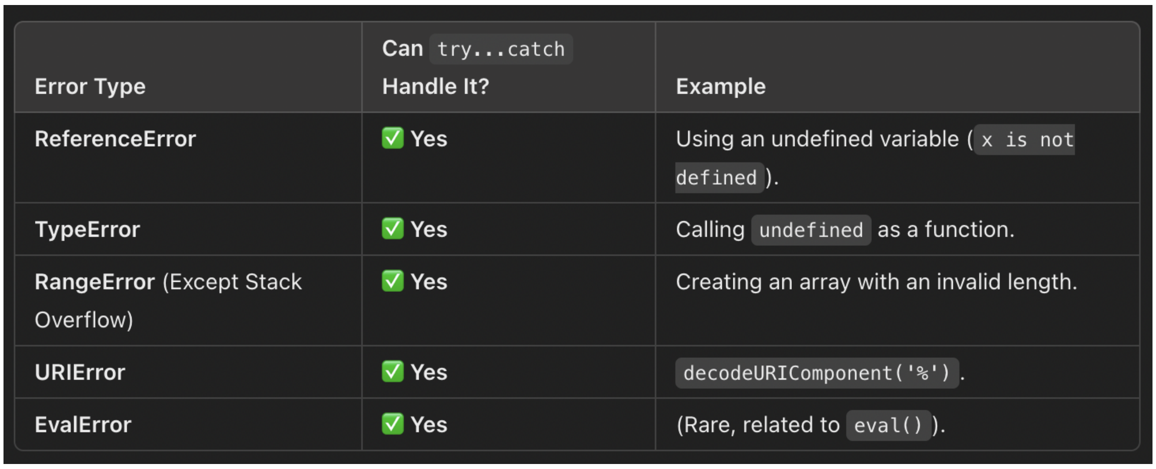
```
try {
    // try_statements
}
catch(error) {
    // catch_statements
}
finally() {
    // codes that gets executed anyway
}
```

▼ Errors that JS cannot catch



Error Type	Can `try...catch` Handle?	Fix/Workaround
SyntaxError	❌ No	Use `eval()` or `new Function()`.
Stack Overflow (`RangeError`)	❌ No	Avoid infinite recursion.
Memory Leaks	❌ No	Improve memory management.
Async Errors (without handling)	❌ No	Use `.catch()` or `try...catch` inside `async`.
Event Listener Errors	❌ No	Use `try...catch` inside the listener.
Network Errors (`fetch()`)	❌ No	Use `.catch()` on the Promise.

▼ Errors that JS can catch



Error Type	Can `try...catch` Handle It?	Example
ReferenceError	✅ **Yes**	Using an undefined variable (`x is not defined`).
TypeError	✅ **Yes**	Calling `undefined` as a function.
RangeError (Except Stack Overflow)	✅ **Yes**	Creating an array with an invalid length.
URIError	✅ **Yes**	`decodeURIComponent('%')`.
EvalError	✅ **Yes**	(Rare, related to `eval()`).