

Error handling

Call stack

- Call stack is the mechanism the JS interpreter uses to keep track of its place in a script that calls multiple functions.
- When a script calls a function, the interpreter adds it to the call stack and then starts carrying out the function. Functions called by that function are added to the call stack higher up and run when their call is reached.
- It could be explained like a list of tasks that JS performs one by one.
- JavaScript is single-threaded, so movement in the call stack is straightforward.

What is error handling? Why is it needed?

- An error is an issue in the code that prevents your program from running normally. Exception is another word for error.
- Error handling involves anticipating, detecting and then resolving problems that can occur during the execution of a program.
- Errors can cause your program to crash. Handling errors where they can possibly arise minimizes the chances of a critical error. A critical error in this context means an error that causes your program to become unusable.

Situations where error handling is useful

- Error handling should be used when working with external data.
- When parsing JSON strings.
- In parts of your code where errors are to be expected.
- When using asynchronous code.
- In situations where you are using features not supported by all browsers.

Common error types

- **Error**: the generic Error constructor creates an error object. Errors are objects in JavaScript.
- **SyntaxError**: occurs when there's a syntactical mistake in the code.
- **ReferenceError**: occurs when a non-existent variable is referenced.
- **TypeError**: occurs when an operation is performed on a value of an unexpected type.
- **RangeError**: occurs when a value is not in the range of allowed values. For example, this can occur when trying to access a non-existent index in an array.

Error object

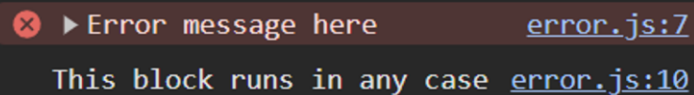
- Error is an object in JavaScript.
- Error messages are displayed in the console when runtime errors occur.
- **Runtime (execution phase)** is the final phase of a computer program's life cycle. It's when the program runs. Runtime errors are errors that occur when the program is running.
- When you're programming in a browser environment, the console should be open most of the time. Try your code often and look for any errors or mistakes in logic.

Try Catch Finally

- Try catch is used to catch runtime errors.
- JavaScript error handling can be systematically done using the try, catch, finally and throw statements.
- Try tries to run a block of code.
- In case of an error, the catch block “catches” it and can display an appropriate error message. The error message can be built-in or user defined.
- The finally block runs regardless of the try and catch blocks.

Using try-catch

```
try {  
  console.log(x); // Accessing a non-existent variable.  
}  
catch {  
  console.error("Error message here");  
}  
finally {  
  console.log("This block runs in any case");  
}
```



✖ ▶ Error message here [error.js:7](#)
This block runs in any case [error.js:10](#)

Throw

- The throw statement is used to generate a custom error object (error message). Current function execution will stop and control is passed to the first catch block in the call stack.
- Showing exception messages is referred to as throwing.
- Throw inside a function terminates function execution.
- When you program JavaScript, you will get errors. Throw is used to handle them.
- Using throw:

```
throw new Error("virhe");
```

```
Error: virhe  
    at Object.<anonymous>
```

Console warning

- A warning is a console message alerting the user (probably developer or tester) about potential issues. It isn't an error in itself.
- Warnings can be outputted by using the console.warn() method.
- Console.warn outputs a warning message to the console.
- Warnings can be used in debugging and to notify of potential issues.
- Just type in the warning message you want to display.

```
console.warn("warning");
```

```
▲ ▶ warning error.js:1  
>
```

Keywords

- Call stack
- Error
- Error object
- Error handling
- Try catch finally
- Throw
- Stack trace
- Stack overflow