Något kommer att gå fel!

- ✓ Allt går inte alltid bra i din applikation. Flera olika typer av fel kan inträffa.
  - Du kan ha gjort misstag i koden.
  - Felaktigt indata kan leda till exekveringsfel.
  - Andra oförutsedda saker.

```
const name = 'Lovelace'

name = 'Hopper' // ERROR: TypeError:  Assignment to constant variable.

console.log(name) // (A)
```

```
OUTPUT
file:///C:\Playground\demo\src\app.js:10
    name = 'Hopper'
         ^

TypeError: Assignment to constant variable.
    at Object.<anonymous> (file:///C:\Playground\demo\src\app.js:10:6)
    at ModuleJob.run (internal/modules/esm/module_job.js:140:23)
    at async Loader.import (internal/modules/esm/loader.js:165:24)
    at async Object.loadESM (internal/process/esm_loader.js:68:5)
```

Resultat då ett undantag inte hanteras!

  - Då ett exekveringsfel inträffar, kastas ett undantag i form av ett objekt och exekveringen avslutas. Satsen A exekveras aldrig!

- ✓ Ett undantag är det objekt, representerande felet, som kastas. Det finns flera olika inbyggda typer av objekt som används vid fel för att skapa undantag.
  - `Error`, `EvalError`, `InternalError`, `RangeError`, `ReferenceError`, `SyntaxError`, `TypeError` och `URIError`.
  - Vanligtvis kommer kod du skriver att kasta `Error` och `TypeError`.

Kasta undantag

- ✓ Ett fel i JavaScript är ett objekt som senare kastas för att hindra fortsatt exekvering av kod.
- ✓ För att skapa objekt som representerar ett fel används det reserverade ordet `new`.

```
const err = new Error('An unexpected error occurred!')
console.log(err.name) // OUTPUT: Error
console.log(err.message) // OUTPUT: An unexpected error occurred!
```

- ✓ Det är inte förrän felobjektet kastas som det skapas ett undantag. För att kasta ett undantag används `throw`.

```
const getDigitSum = function (value) {
  if (typeof value !== 'number') {
    throw new TypeError('The argument passed must be a Number.')
  }

  // TODO: Sum all the digits and return the sum
}


getDigitSum('This is not a number')
```

```
OUTPUT
file:///C:/Playground/demo/src/app.js:16
    const err = new TypeError('The argument passed must be a Number.')
                ^

TypeError: The argument passed must be a Number.
    at getDigitSum (file:///C:/Playground/demo/src/app.js:16:17)
    at file:///C:/Playground/demo/src/app.js:24:1
    at ModuleJob.run (internal/modules/esm/module_job.js:140:23)
    at async Loader.import (internal/modules/esm/loader.js:165:24)
    at async Object.loadESM (internal/process/esm_loader.js:68:5)
```

Resultat då undantaget inte hanteras!

Fånga undantag

- ✓ Kod som kan leda till att ett undantag kastas placeras i ett `try`-block, som följs av ett...
- ✓ ...`catch-block` innehållande kod som ska exekveras om ett undantag har kastats.
- ✓ Kod placerad i `finally-blocket` körs alltid.

```
const getDigitSum = function (value) {
  if (typeof value !== 'number') {
    const err = new TypeError('The argument passed must be a Number.')
    throw err
  }

  // TODO: Sum the digits and return the value
}

try {
  const result = getDigitSum('This is not a number')
  console.log(`The sum of the digits is ${result}.`)
} catch (err) {
  console.error(err.message)
} finally {
  // Code in this block will always be executed'
}
```

Kan kombineras enligt:
- `try-catch`
- `try-finally`
- `try-catch-finally`

OUTPUT
The argument passed must be a Number.