***Bugs and Errors***

Debuggen is moeilijk, dus als je code zo slim mogelijk probeert op te schrijven, wordt debuggen 2 keer zo moeilijk.

Verschillende bugs:

- Waarbij gedachtes worden verward

- Waarbij er fouten ontstaan bij het omzetten van fouten in code 🡪 Moeilijker te fixen

Javascript haalt fouten er vrijwel nooit uit, en de fout kan zich pas veel later in je code openbaren. **Debuggen** is dus essentieel.

Bij **“use strict”** zal JavaScript iets meer fouten herkennen. Altijd handig om te gebruiken dus 🡪 linter haalt deze meeste dingen er allang uit.

Het is handig om het type waarde wat er uit je functie moet in- en uitkomen weet, zodat je hiermee fouten kan voorkomen. Zoals dit:

*// (VillageState, Array) → {direction: string, memory: Array}*

*function goalOrientedRobot(state, memory) {*

*// ...*

*}*

Met **TypeScript** kan je aangeven welk type waarde je verwacht en dit controleren.

**Test runners** = software die je functies en methods testen en uitleg geven. 🡪 OOP werkt het beste.

Je kan debuggen door:

- Na te denken wat je programma stap voor stap doet door na te denken en console.log te plaatsen om dat te verifiëren.

- Debugger gebruiken. In je browser of als keyword in JS, JavaScript pauzeert dan en laat zien wat er allemaal aan de hand is op dat moment.

- Iets te plaatsen (zoals een if statement) in je code waarbij je rekening houdt met een foutieve waarde en vervolgens iets teruggeeft waarbij jij weet dat het daar fout is gegaan, zoals ‘failed: true’.

**Exception(s handeling)** = wanneer een code tegen een probleem aanloopt, uit alle bijbehorende functies en callers gaat. Ook wel ***unwinding the stack*** genoemd.

Met throw kan je een exception aanduiden. Deze kan je vervolgnes **catchen** door het in een **try** block te wrappen. De catch evalueert het vervolgens en gaat door of niet. Voorbeeld:

*function promptDirection(question) {*

*let result = prompt(question);*

*if (result.toLowerCase() == "left") return "L";*

*if (result.toLowerCase() == "right") return "R";*

*throw new Error("Invalid direction: " + result);*

*}*

*function look() {*

*if (promptDirection("Which way?") == "L") {*

*return "a house";*

*} else {*

*return "two angry bears";*

*}*

*}*

*try {*

*console.log("You see", look());*

*} catch (error) {*

*console.log("Something went wrong: " + error);*

*}*

Wanneer de error wordt gerunned, wordt er informatie over de callstack verzamelt, de **stack trace.** Dit wordt opgeslagen in de stack property.

**Finally** kan ook in een try statement. Dit betekent: het maakt niet uit wat er gebeurd, run deze code na het proberen te runnen van het try block.