Fejlhåndtering

`strict mode`

- JavaScript er med sin automatisk typekonvertering og manglende typetjek ret tolerant over for fejl
- Brug af strict mode kan fange lidt flere fejl, så som manglende erklæring af variable og brug af en funktions globale this
- `'use strict';` skal skrives øverst i filen eller i starten af en funktion

```
// strict.js
'use strict';

function f() { x = 1; }
f(); // => ReferenceError: x is not defined

function g(x) { this.x = x; }
g(); // => TypeError: Cannot set property 'x' of undefined
```

Error som exception

- JavaScript definerer en generel Error type og nogle mere specifikke fejltyper: ReferenceError, TypeError og SyntaxError mfl.
- Sætningen throw kaster en exception, der kan være et vilkårligt udtryk
- Men det anbefales at bruge Error – eller en specialisering af denne – som exception, da den også viser et stack trace

```
// error.js
throw 'Dette er en string som exception';
// => Dette er en string som exception
throw Error('Dette er en Error som exception');
// => Error: Dette er en Error som exception
//     at Object.<anonymous>
//          (C:\Users\ed\WebstormProjects\Fejlhåndtering\error.js:3:7)
//     at ...
```

Validering af parametre

- Da en funktions parametre er uden type, bør parametrene valideres i starten af funktionen
- Er funktionen "overloaded", skal den aktuelle variant desuden identificeres

```
// validering.js
function max(...elementer) {
    if (elementer.length === 0) throw Error('Ingen data');
    let type = typeof elementer[0];
    if (type !== 'number' && type !== 'string') throw Error('Forkert type');
    if (!elementer.every(e => typeof e === type)) throw Error('Ej samme type');
    if (type === 'number')
        return elementer.reduce((a, e) => a>e?a:e);
    else  // type == 'string'
        return elementer.reduce((a, e) => a.localeCompare(e)>0?a:e);
}
```

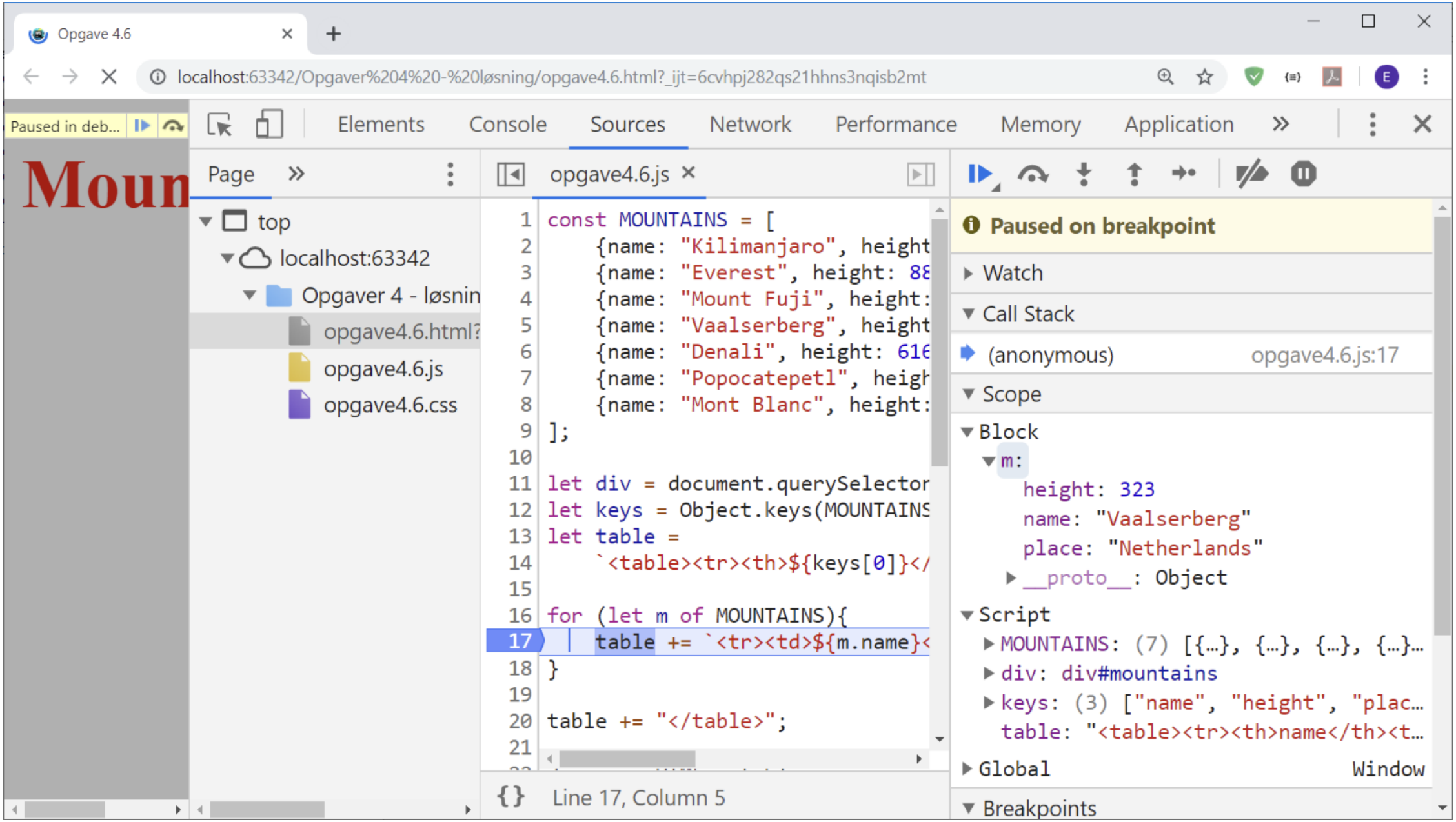
```
// validering.js
function max(...elementer) {
    if (elementer.length === 0) throw Error('Ingen data');
    let type = typeof elementer[0];
    if (type !== 'number' && type !== 'string') throw Error('Forkert type');
    if (!elementer.every(e => typeof e === type)) throw Error('Ej samme type');
    if (type === 'number')
        return elementer.reduce((a, e) => a>e?a:e);
    else  // type == 'string'
        return elementer.reduce((a, e) => a.localeCompare(e)>0?a:e);
}

console.log(max(1, 3, 2)); // => 3
console.log(max('A', 'b', 'C')); // => C
console.log(max('x')); // => x
console.log(max()); // => Error: Ingen data
console.log(max(max, null)); // => Error: Forkert type
console.log(max(1, 'tre', 2)); // => Error: Ej samme type
```

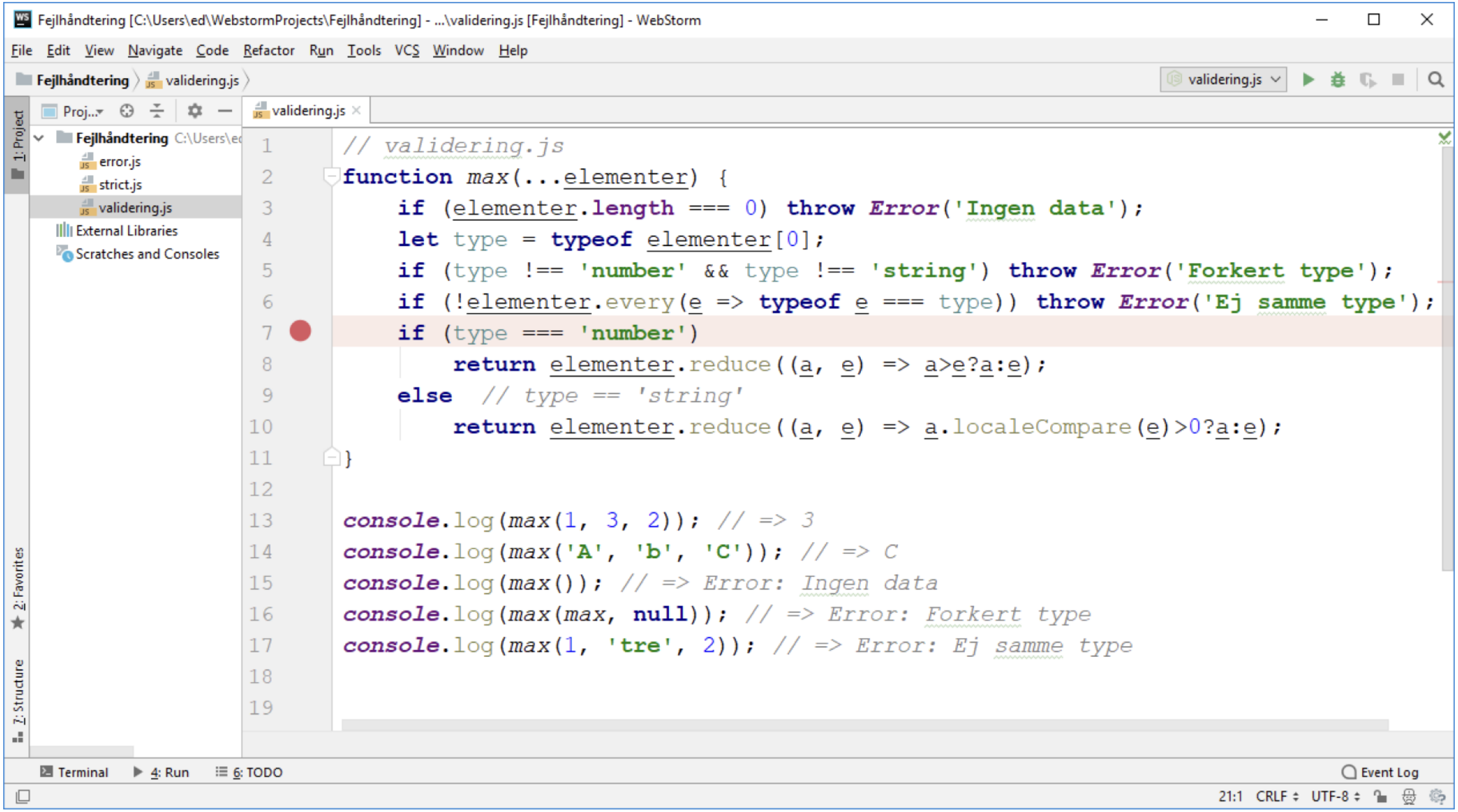
Validering af function-parametre og arrays

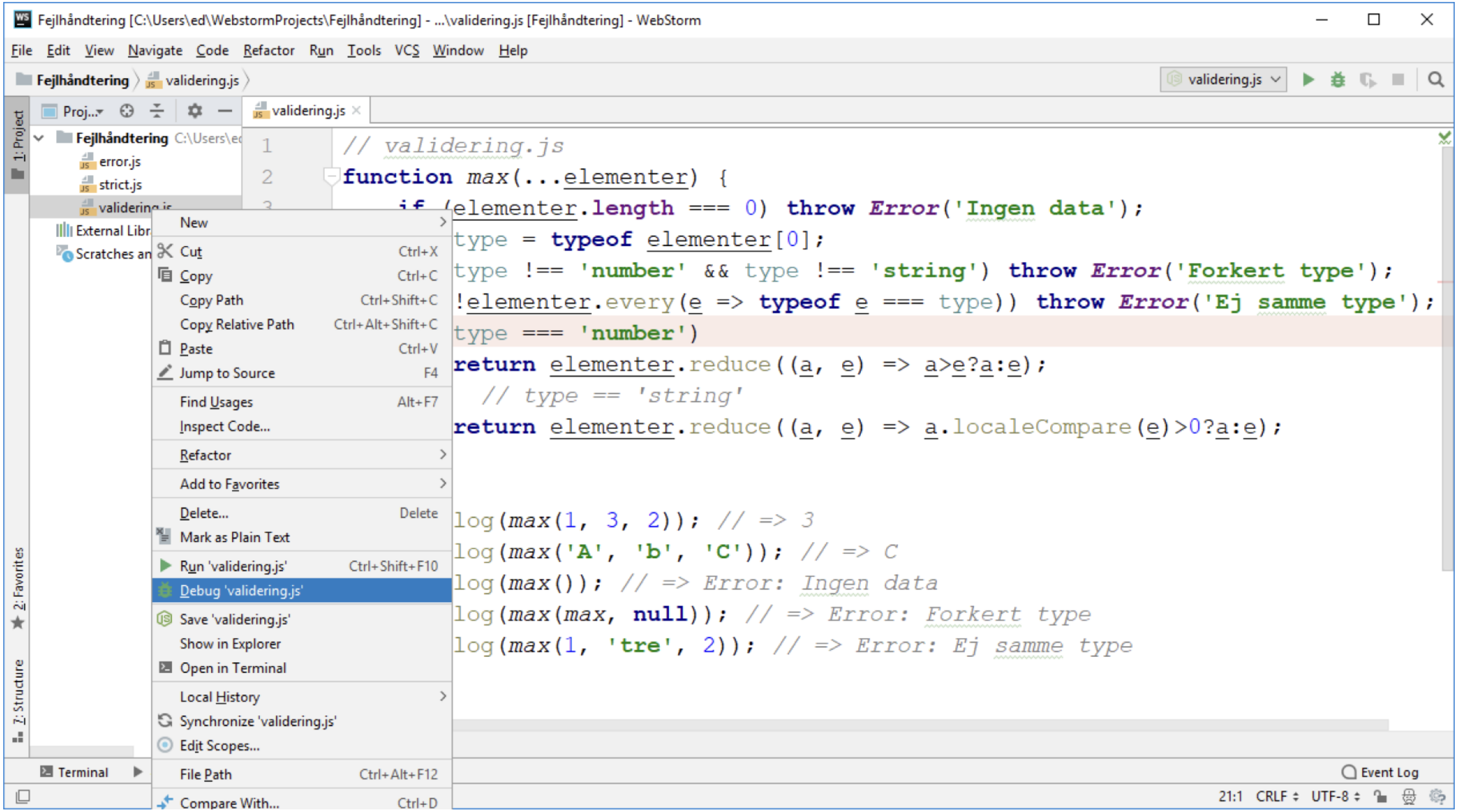
- arguments er et array af argumenterne til et kald af en function. arguments.length kan bruges til at checke antallet. (https://www.w3schools.com/js/js_function_parameters.asp)
- typeof kan bruges til at checke, om et argument er en function (typeof a == 'function')
- f.length, hvor f er en function, giver antal parametre til metoden (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Function/length)
- (parametre er de parametre, funktionen forventer (de formelle parametre), argumenter er de parametre, der reelt bliver overført i et konkret kald (de aktuelle parametre))
- Array.isArray(x) checker, om x er et array

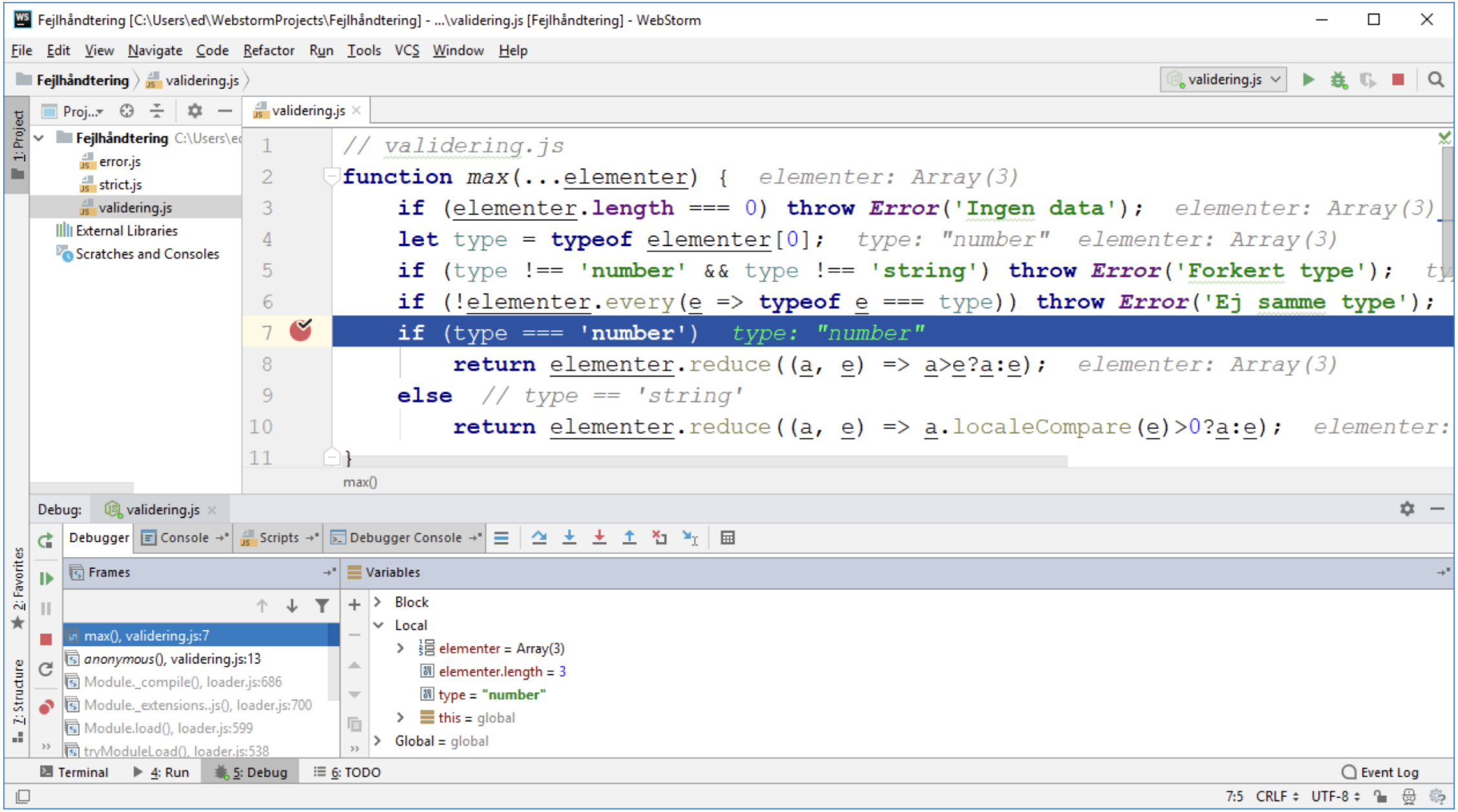
Debugging

- En debugger gør det lettere at finde fejl
- Der kan sættes *breakpoints* og *watches*, og koden udføres trin for trin, så det er let at se, hvordan variable ændrer værdi og hvor programmet bevæger sig hen
- En debugger giver et godt overblik og er mere effektiv end at indsætte en række `console.log()` sætninger


Opgave 4.6
localhost:63342/Opgaver%204%20-%20løsning/opgave4.6.html?_ijt=6cvhpj282qs21hhns3nqisb2mt
Paused in deb...
Moun
Elements
Console
Sources
Network
Performance
Memory
Application
Page
top
localhost:63342
Opgaver 4 - løsnin
opgave4.6.html?
opgave4.6.js
opgave4.6.css
opgave4.6.js
1 const MOUNTAINS = [
2 {name: "Kilimanjaro", height
3 {name: "Everest", height: 88
4 {name: "Mount Fuji", height:
5 {name: "Vaalserberg", height
6 {name: "Denali", height: 616
7 {name: "Popocatepetl", heigh
8 {name: "Mont Blanc", height:
9];
10
11 let div = document.querySelector
12 let keys = Object.keys(MOUNTAINS
13 let table =
14 `<table><tr><th>${keys[0]}</
15
16 for (let m of MOUNTAINS){
17 table += `<tr><td>${m.name}<
18 }
19
20 table += "</table>";
21
{} Line 17, Column 5
Paused on breakpoint
Watch
Call Stack
(anonymous) opgave4.6.js:17
Scope
Block
m:
height: 323
name: "Vaalserberg"
place: "Netherlands"
__proto__: Object
Script
MOUNTAINS: (7) [{…}, {…}, {…}, {…}…
div: div#mountains
keys: (3) ["name", "height", "plac…
table: "<table><tr><th>name</th><t…
Global Window
Breakpoints


Fejlhåndtering [C:\Users\ed\WebstormProjects\Fejlhåndtering] - ...\validering.js [Fejlhåndtering] - WebStorm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Fejlhåndtering
validering.js
validering.js
1: Project
Proj...
Fejlhåndtering C:\Users\ed
error.js
strict.js
validering.js
External Libraries
Scratches and Consoles
1 // validering.js
2 function max(...elementer) {
3 if (elementer.length === 0) throw Error('Ingen data');
4 let type = typeof elementer[0];
5 if (type !== 'number' && type !== 'string') throw Error('Forkert type');
6 if (!elementer.every(e => typeof e === type)) throw Error('Ej samme type');
7 if (type === 'number')
8 return elementer.reduce((a, e) => a>e?a:e);
9 else // type == 'string'
10 return elementer.reduce((a, e) => a.localeCompare(e)>0?a:e);
11 }
12
13 console.log(max(1, 3, 2)); // => 3
14 console.log(max('A', 'b', 'C')); // => C
15 console.log(max()); // => Error: Ingen data
16 console.log(max(max, null)); // => Error: Forkert type
17 console.log(max(1, 'tre', 2)); // => Error: Ej samme type
18
19
2: Favorites
7: Structure
Terminal
4: Run
6: TODO
Event Log
21:1 CRLF UTF-8


Fejlhåndtering [C:\Users\ed\WebstormProjects\Fejlhåndtering] - ...\validering.js [Fejlhåndtering] - WebStorm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Fejlhåndtering
validering.js
validering.js
1: Project
2: Favorites
7: Structure
Proj...
Fejlhåndtering C:\Users\ed
error.js
strict.js
validering.js
External Libr
Scratches an
// validering.js
function max(...elementer) {
if (elementer.length === 0) throw Error('Ingen data');
type = typeof elementer[0];
type !== 'number' && type !== 'string') throw Error('Forkert type');
!elementer.every(e => typeof e === type)) throw Error('Ej samme type');
type === 'number')
return elementer.reduce((a, e) => a>e?a:e);
// type == 'string'
return elementer.reduce((a, e) => a.localeCompare(e)>0?a:e);
log(max(1, 3, 2)); // => 3
log(max('A', 'b', 'C')); // => C
log(max()); // => Error: Ingen data
log(max(max, null)); // => Error: Forkert type
log(max(1, 'tre', 2)); // => Error: Ej samme type
New
Cut Ctrl+X
Copy Ctrl+C
Copy Path Ctrl+Shift+C
Copy Relative Path Ctrl+Alt+Shift+C
Paste Ctrl+V
Jump to Source F4
Find Usages Alt+F7
Inspect Code...
Refactor
Add to Favorites
Delete... Delete
Mark as Plain Text
Run 'validering.js' Ctrl+Shift+F10
Debug 'validering.js'
Save 'validering.js'
Show in Explorer
Open in Terminal
Local History
Synchronize 'validering.js'
Edit Scopes...
File Path Ctrl+Alt+F12
Compare With... Ctrl+D
Terminal
Event Log
21:1 CRLF UTF-8


Fejlhåndtering [C:\Users\ed\WebstormProjects\Fejlhåndtering] - ...\validering.js [Fejlhåndtering] - WebStorm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Fejlhåndtering
validering.js
validering.js
1: Project
Proj...
Fejlhåndtering C:\Users\ed
error.js
strict.js
validering.js
External Libraries
Scratches and Consoles
1 // validering.js
2 function max(...elementer) { elementer: Array(3)
3 if (elementer.length === 0) throw Error('Ingen data'); elementer: Array(3)
4 let type = typeof elementer[0]; type: "number" elementer: Array(3)
5 if (type !== 'number' && type !== 'string') throw Error('Forkert type'); ty
6 if (!elementer.every(e => typeof e === type)) throw Error('Ej samme type');
7 if (type === 'number') type: "number"
8 return elementer.reduce((a, e) => a>e?a:e); elementer: Array(3)
9 else // type == 'string'
10 return elementer.reduce((a, e) => a.localeCompare(e)>0?a:e); elementer:
11 }
max()
Debug: validering.js
Debugger
Console
Scripts
Debugger Console
Frames
Variables
max(), validering.js:7
anonymous(), validering.js:13
Module._compile(), loader.js:686
Module._extensions..js(), loader.js:700
Module.load(), loader.js:599
tryModuleLoad(), loader.js:538
Block
Local
elementer = Array(3)
elementer.length = 3
type = "number"
this = global
Global = global
2: Favorites
7: Structure
Terminal
4: Run
5: Debug
6: TODO
Event Log
7:5 CRLF UTF-8