Webscripting

Hoofdstuk 8

Bugs and errors

DE HOGESCHOOL
MET HET NETWERK

Hogeschool PXL – Elfde-Liniestraat 24 – B-3500 Hasselt
www.pxl.be - www.pxl.be/facebook

use strict

- ‘use strict’ ofwel bovenaan file ofwel in function
- nut: global bindings vermijden

```
// global binding:
name = 'tim';
// ipv let name = 'tim';
console.log(name);  // tim
console.log(global.name); // tim
```

De variabele name wordt in het global object geplaatst.

global object bevat voorgedefinieerde functions / variables (e.g. setInterval)

Voor browser global object = window
Voor nodejs global object = global

use strict

- ‘use strict’ ofwel bovenaan file ofwel in function
- nut: aanmaken van global bindings vermijden

```
'use strict'
name = 'tim';
console.log(name);  // tim

// foutmelding:
// name = 'tim';
//      ^
// ReferenceError: name is not defined
```

use strict

- ‘use strict’ ofwel bovenaan file ofwel in function
- nut: aanmaken van global bindings vermijden

```
function canYouSpotTheProblem() {
    'use strict'
    for (counter = 0; counter < 10; counter++) {
// ipv for (let counter = 0; ...
        console.log("Happy happy");
    }
}
canYouSpotTheProblem();

// ReferenceError: counter is not defined
// for (counter = 0; counter < 10; counter++) {
//                 ^
// ReferenceError: counter is not defined
```

use strict

this in een functie

zonder use strict

- this ~ object indien de function deel uitmaakt v. object
- this ~ global object indien de functie niet deel uitmaakt v. object

met use strict

- this ~ object indien de function deel uitmaakt v. object
- this ~ undefined indien de functie niet deel uitmaakt v.e. object

```
function Person(name) {
    this.name = name;
}

let ferdinand = Person("Ferdinand");
// eigenlijk wou ik new Person("Ferdinand") om een object
// aan te maken
// Person wordt aangeroepen, this ~ global object
console.log(name); // Ferdinand
```

```
'use strict'
function Person(name) {
    this.name = name;
}
let ferdinand = Person("Ferdinand");
// eigenlijk wou ik = new Person("Ferdinand") om een
object // aan te maken
console.log(name);

// foutmelding
// function Person(name) { this.name = name; }
//                                  ^
// TypeError: Cannot set property 'name' of undefined
```

Errors: try … catch

```
try {
    ...
    // Probeer deze code uit te voeren
    // Indien een error opgeworpen wordt, staakt de executie
    // en wordt de code in de catch block uitgevoerd
    ...
} catch (error) {
    ...
    // catch block
    ...
}
```

Errors: try … catch … finally

```
try {
    ...
    // Probeer deze code uit te voeren
    // Indien een error opgeworpen wordt, staakt de executie
    // en wordt de code in de catch block uitgevoerd
    ...
} catch (error) {
    ...
    // catch block
    ...
} finally {
    ...
    // finally block wordt altijd uitgevoerd, ongeacht of er
    // een exception opgeworpen wordt of niet
}
```

Errors: throw

throw: werp een error op
verdere uitvoer wordt gestaakt

```
'use strict'
function invert(number){
    if (typeof number != 'number'){
        throw new Error("not a number");
    }
    if (number == 0){
        throw new Error("division by 0");
    }
    return 1 / number;
}

try{
    let result = invert(0);
    console.log(`result = ${result}`);
} catch( error ) {
        console.log(`Exception: ${error.message}`);
}
//Exception: division by 0
```

Errors

```
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);
}
```

Errors: specifieke errors opvangen

```
'use strict'
class InputError extends Error {}
class ArithmeticError extends Error {}

function invert(number){
    if (typeof number != 'number'){
        throw new InputError("not a number");
    }
    if (number == 0){
        throw new ArithmeticError("division by 0");
    }

    return 1 / number;
}
```

Errors: specifieke errors opvangen

```
try{
    let result = invert(0);
    console.log(`result = ${result}`);
} catch( error ) {
    if (error instanceof InputError){
        console.log(`InputError: ${error.message}`);
    } else if (error instanceof ArithmeticError){
        console.log(`ArithmeticError: ${error.message}`);
    }
}
```

Besluit

use strict: global bindings vermijden

try … catch
    probeer een stuk code uit te voeren
    indien error: executie v. try wordt gestaakt
                   catch wordt uitgevoerd

throw: werp een error op